A Study of Mobility Models in Mobile Surveillance Systems

by

Yun-Qian Miao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

Yun-Qian Miao

I understand that my thesis may be made electronically available to the public.

Yun-Qian Miao

Abstract

This thesis explores the role mobile sensor's mobility model and how it affects surveillance system performance in term of area coverage and detection effectiveness. Several algorithms which are categorized into three types, namely, fully coordinated mobility, fully random mobility and emergent mobility models are discussed with their advantages and limitations.

A multi-agent platform to organize mobile sensor nodes, control nodes and actor nodes was implemented. It demonstrated great flexibility and was favourable for its distributed, autonomous and cooperative problem-solving characters.

Realistic scenarios which are based on three KheperaIII mobile robots and a model which mimics Waterloo regional airport were used to examine the implementation platform and evaluate performance of different mobility algorithms. Several practical issues related to software configurations and interface library were addressed as by-products.

The experimental results from both simulation and real platform show that the area coverage and the detection effectiveness vary with applying different mobility models. Fully coordinated model's super efficiency comes with carefully task planning and high requirements of sensor navigational accuracy. Fully random model is the least efficient in area coverage and detection because of the repetitive searching of each sensor and among sensors.

A self-organizing algorithm named anti-flocking which mimics solitary animal's social behaviour was first proposed. It works based on quite simple rules for achieving purposeful coordinated group action without explicit global control. Experimental results demonstrate its attractive target detection efficiency in term of both detection rate and detection time while providing desirable features such as scalability, robustness and adaptivity. From the simulation results, the detection rate of the anti-flocking model increases by 36.5% and average detection time decreases by 46.2% comparing with the fully random motion model. The real platform results also reflect the superior performance improvement.

Acknowledgements

I would express my sincere thanks to my supervisor Professor Mohamed S. Kamel for his technical, financial and moral support.

My co-supervisor Dr. Alaa Khamis guided me through every step of this research and reviewed my draft materials. I would like to thank him for giving generously of his time and efforts.

Many thanks are also due to my thesis readers Professor Fakhreddine Karray and Professor Eihab Abdel-Rahman for taking the time and assessing my work within a short time frame.

I also want to express my thanks to the Robotics and Autonomous System (RAS) focus group members: Bahador Khaleghi, Manglai Zhou,Ahmed Elmogy, Allaa Hilal, Mehrdad Gangeh, Rodrigo Araujo, Karim El-Rayes, for their constructive feedback during our regular group meeting.

Finally, the author appreciates the financial support provided by Government of Ontario through the project of MUSES_SECRET.

Dedication

This is dedicated to my wife Xiaochun.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Recent events, including major terrorist attacks such as the September 11, 2001 on America, have led to an increased demand for public security. One of the most prominent ongoing reactions is to enhance surveillance operations. The increasing need for intelligent surveillance in public, military and commercial applications makes surveillance systems becoming one of the main application domains and attracting a lot of research interests from multiple disciplines such as signal processing, communications, control theory, software engineering, management, and socio-ethical studies [32, 55, 40].

Surveillance systems collect data and information about different aspects of objects and events in a given Area Of Interest (AOI) and fuse them in order to provide a complete picture of the situation of interest. This picture contains a representation of the area under surveillance (e.g., an airport) that essentially displays information about red objects (adversary, non-cooperative), blue objects (own and friendly), white objects (neutral, usually cooperative), and environmental conditions [49, 38].

There are many factors that increase the complexity of surveillance operations such as high tempo, high density in certain environments such as school campuses, shopping centers, airports, etc. and the collateral damages in critical applications with dense activity such as military operations, border patrol, coastal surveillance and reconnaissance of secured rural areas and cities.

Fully leveraging information superiority and achieving robust cooperation between distributed surveillance units through network-centric capabilities provide a promising solution to overcome these difficulties. Last decade has witnessed intense research activities in stationary sensor networks [11, 29, 53]. Mobile sensor networks started to attract the

attention of many researchers as an enabling technology for intelligent mobile surveillance systems [17, 18, 66, 37, 52]. The findings of the research activities helped in tackling and understanding some of the challenging problems of mobile surveillance systems that are still open. These research problems include, but are not limited to, mobile sensor deployment, task allocation, multisensor management, cooperative object detection and tracking, decentralized data fusion and interoperability and accessibility of system nodes.

The problem of mobile sensor deployment addresses how to optimally deploy a set of mobile sensors in a given AOI in such a way that achieves a certain objective. This objective can be, but is not limited to, maximizing the area and/or target coverage [35, 21], improving secure communications between the mobile nodes (maximizing the radio coverage) [22, 3, 33] or improving the detection rate over agile targets [25]. Relatively little research has explored mobile sensors mobility model for area coverage and target detection.

For this purpose, this thesis will study different mobility model of mobile sensors that coordinates these multiple distributed sensing nodes in order to accomplish surveillance tasks in an efficient and adaptive way in dynamic environment.

1.2 Objectives

Comparing with surveillance system utilizing stationary sensors, performance of mobile sensor network depends not only on the initial network configurations, but also on the mobility models of the mobile sensors. Adding the dimension of mobility of sensor network for surveillance raises some new challenges such as how to evaluate the effectiveness of the mobile surveillance system?, how to organize the mobility strategy that can maximum these metrics of effectiveness?, how are these algorithms robust and adaptable in dynamic environments?

To answer above important questions, this research work will put focuses on:

- Studying different mobility models of mobile sensors that are applied in mobile surveillance system;
- Exploring mission-suitable self-organized algorithm that can effectively coordinate multiple mobile sensors;
- Setting up a set of performance evaluation metrics to examine the effectiveness of different mobility models;
- Developing a mobile surveillance system platform that is based on multi-agent architecture, and using this platform to test different algorithms related to mobile sensor's mobility models.

1.3 Contributions

This research activity and its new findings and new achievements have contributed to both the academic world and application fields in the following aspects:

- A multi-agent platform to organize mobile sensor nodes, control nodes and actor nodes was implemented. It demonstrated great flexibility and was favourable for its distributed, autonomous and cooperative problem-solving characters;
- The relationship between mobile sensor's mobility models, area coverage and detection efficiency were analyzed and examined. The advantages and constraints of different algorithms were well studied.
- A self-organizing algorithm named anti-flocking which mimics solitary animal's social behaviour was first proposed. Experimental results demonstrate its attractive target detection efficiency while providing desirable features such as scalability, robustness and adaptivity;
- A new version of Java client interface library (to be released as free GNU software) was developed. It bridges present version of mobile robot control software package (Player/Stage), and will benefit both academic and industrial world;

1.4 Thesis structure

The remainder of this thesis is organized as follows. Chapter 2 discusses the concept, the components, and the state-of-the-art of the mobile surveillance system.

Chapter 3 explores three categories of mobility models of mobile sensors, fully coordinated, fully random, and emergent control strategy. Several practical algorithms are given in each category.

Chapter 4 discusses the commonly used performance evaluation metrics in mobile surveillance systems, and derives a set of MOE (measures of effectiveness) that is used in this work.

Chapter 5 gives the experimental setups, experimental results and discussion.

Chapter 6 summarizes this work with conclusions and future directions.

Appendix A describes the different components of the implementation environment such as Player/Stage as mobile robot driver and simulation platform, JADE as agent management platform, and JavaClient as a bridge between Java agent and Player server.

Appendix B describes connection setups and software configuration of the KheperaIII robot.

Finally, Appendix C provides software user guide that describes how to use the developed system.

Chapter 2

Mobile Surveillance Systems

This chapter starts by reviewing the evolution history of surveillance systems first. As one of the highly potential research directions and application trends, the mobile surveillance system will be introduced by describing its architecture, functional components, and related concepts. The concluding remarks summarize challenges of mobile surveillance system and the focus of this thesis.

2.1 Surveillance operations

Surveillance operations [32, 65] include the timely detection, localization, recognition and identification of objects and events, their relationships, activities, and plans, in a given Area of Interest (AOI). In application domains such as security or defense, this process is referred to as picture compilation, the aim of which is to generate a representation of the area under surveillance (e.g., an airport) based on data and information from a variety of sources. The compiled picture essentially displays information about red objects (adversary, non-cooperative), blue objects (own and friendly), white objects (neutral, usually cooperative), and environmental conditions. The following are few factors that further define the complexity of surveillance operations:

- High tempo: The limited time available to understand the impact of the events on the mission at hand and to react to them. High tempo imposes the requirement that critical (potential red) objects be detected as early as possible so as to provide more reaction time to the human decision makers.
- High density: Certain environments, such as school campuses, shopping centers, airports, etc., exhibit significant congestion with dense activity (e.g., commercial,

educational and recreational traffic) as compared to open uncrowned environments. High density necessitates increased effort on the part of the system to gain and maintain situation awareness, thus distracting it from focusing on critical objects.

- Collateral Damage: In critical applications, high density also imposes a non-uniform environment that cannot be pictured as a confrontation between blue (friendly) objects and red (enemy or undesirable) objects. Blue and red, as well as neutral (white), objects are interspersed and overlapping, presenting a highly complex challenge with respect to discerning one type of object from another. This situation increases the risk of undesirable effects, e.g., casualties.

2.2 Surveillance systems

Valera and Velastin classify the history of surveillance system into three generations according technological advancement evolution routine [55]. Table 2.1 summarizes the three generation intelligent surveillance systems, outlining the advantages, problems, and current research topics.

The technological evolution of intelligent surveillance systems started with analogue Closed-Circuit Television (CCTV) systems. Even later digital CCTV systems are used for images capture and storage, the whole system consists of a number of sensors, mostly cameras, located in multiple remote locations and connected to a set of monitors, usually placed in a single control room. The tasks of target identification and decision making are on the full responsibility of the human operator. This is the first generation of surveillance system in common.

The second generation of surveillance system is based on the creation of algorithms for automatic real-time detection events that aids the human user to recognize the events and make proper reactions. This stage is known as semi-automatic systems.

The third generation of surveillance system, which is the field of growing research interest, refers to large scale, fully integrated, high intelligent, and distributed processing of surveillance. Based on end-user requirements, the system main goals are to provide human user accuracy picture compilation, good scene understanding, and aiding decision making.

2.3 Multi-sensor surveillance systems

Many researchers have revealed that applying multiple sensors can bring extra benefits than single one could reach [15, 54]. First, the robustness and reliability are of the most

1st generation:	
Techniques	Analogue CCTV systems
Advantages	They give good performance in some situations Mature technology
Problems	Use analogue techniques for image distribution and storage
Current Research	- Digital versus analogue - Digital video recording - CCTV video compression
2nd generation:	
Techniques	Automated visual surveillance by combining computer vision technology with CCTV systems
Advantages	Increase the surveillance efficiency of CCTV systems
Problems	Robust detection and tracking algorithms required for behavioural analysis
Current Research	- Real-time robust computer vision algorithms - Automatic learning of scene variability and patterns of behaviours - Bridging the gap between the statistical analysis of a scene and producing natural language interpretations
3rd generation:	
Techniques	Automated wide-area surveillance system
Advantages	More accurate information as a result of combining different kind of sensors Distribution
Problems	- Distribution of information (integration and communication) - Design methodology - Moving platforms, multi-sensor platforms
Current Research	- Distributed versus centralised intelligence - Data fusion - Probabilistic reasoning framework - Multi-camera surveillance techniques

Table 2.1: Summary of technical evolution of intelligent surveillance systems(from Valera and Velastin [55])

desired important features in the surveillance system, where system robustness are achieved via multi-sensors, while a single mobile sensor cannot reach very high standard of reliability, especially for critical tasks involving hazardous and dynamic environment where the potential of single node failure cannot be ignored and system reliability are highly desired.

A second reason of applying multiple sensors is some tasks inherently need multiple sensors to perform. For example, the task of positioning a target in two dimensions environment need three sensors to measure the distance at the same time, if each one can only measure the distance to the target.

The third reason is multiple sensors can improve the performance of tasks. A group of sensors with different modality can be seen as different measurement modals. Multimodal data fusion is proved to be giving more accuracy and reliable result [46]. In [46], the distributed cooperative Kalman Filter is discussed to have a much accurate result in mobile target estimation. Furthermore, some tasks are easier to be done by multi-sensors while it will be difficult or time consuming for a single one. For example, several cameras with different viewing angles can take images of object with better covering and can generate 3D modal of the target.

2.4 Applications of multi-sensor surveillance

As increasing demand for public safety and security, multi-sensor surveillance system has received particular research and development attention, especially in the following areas:

- Public security applications: a network of multi-sensors used to detect threats, hazardous materials, or any phenomena of importance to public security, such as airports [57], maritime environments [2], railways, subway [31], highways traffic [5, 63], parking lots [39], shopping mall [12], and campus etc.
- Infrastructure health monitoring: creating effective monitoring solutions for bridges and other civil structures. Using Wi-Fi sensing nodes, system is able to provide a quickly deployable solution that can be remotely programmed and changed as needed [59].
- Health care: by connecting a wide range of medical devices to computers and remote services, the multi-sensor architecture allows healthcare providers to create powerful applications for monitoring and intervention. It can also be used to enable medicine reminder and compliance solutions.
- Surveillance in military applications [6, 42].

2.4.1 An example: MUSES-SECRET project

Multimodal- SurvEillance System for SECurity-RElaTed Applications (MUSES_SECRET) project [49] is an ORF-RE project aims at developing and commercializing new multimodal (video and infrared, voice and sound, RFID and perimeter intrusion) intelligent sensor surveillance technologies for the timely identification of human intent and threat assessment in high security-risk dynamic environments involving human crowds, located at places such as school campuses, shopping centers, airports, etc.. Fig.2.1 demonstrates a typical surveillance operation scenario that comprises by multimodal sensing nodes.

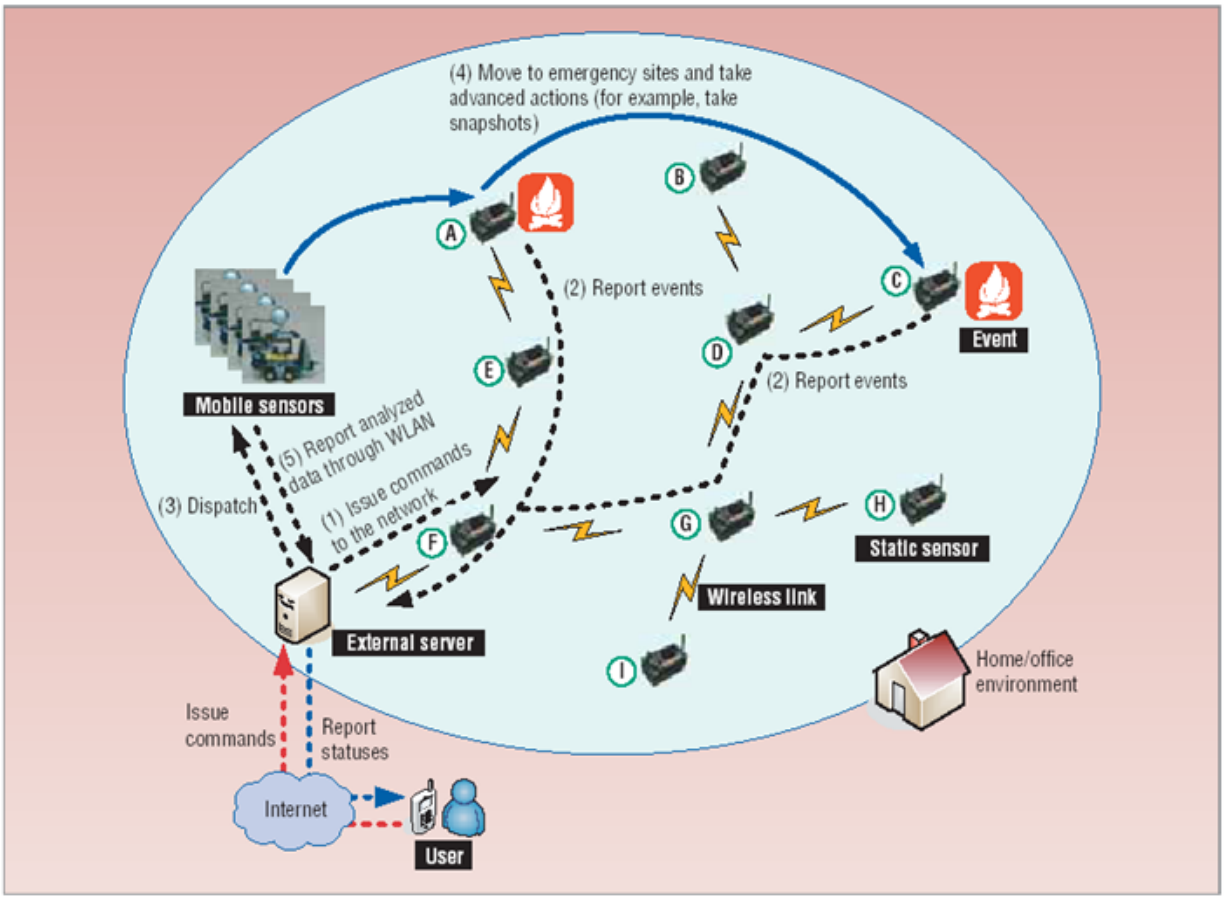

Figure 2.1: Multimodal multi-sensor surveillance using static and mobile Sensors (from Tseng et al. [54]).

Another research objective of this project is automatic recognition of people and their activities. Automatic recognition of people and their activities has very important social implications, because it is related to the extremely sensitive topic of civil liberties. Society

needs to address this issue of automatic recognition and find a balanced solution that is able to meet its various needs and concerns. In the post 9/11 period, population security and safety considerations have given rise to research needs for identification of threatening human activities and emotional behaviours.

Multi-sensor data fusion techniques are also being investigated for the dynamic integration of the multi-thread flow of information provided by the heterogeneous network of surveillance sensors into a coherent multimodal model of the monitored human crowd. Real-time image processing and computer-vision algorithms will be studied for the identification, tracking and recognition of gait and other relevant body-language patterns of the human agents who can be deemed of interest for security reasons. Real-time signal processing algorithms will be designed for the identification and evaluation of environmental and human behaviour multimodal parameters (such as human gait, gestures, facial emotions, human voice, background sound, ambient light, etc.) that provide the contextual information for the specific surveillance activity.

In this project, a multidisciplinary, context-aware, situation-assessment system, including human behaviour, cognitive psychology, multicultural sociology, learning systems, artificial intelligence, distributed multimedia and software design elements, will be ultimately developed for the real-time evaluation of the activity and emotional behaviour of the human subjects identified as being potentially of security interest in the monitored dynamic environment. The resulting system should provide efficient multi granularity-level function-specific feedback for the human users who are the final assessors and decision makers in the specific security monitoring situation.

2.5 Mobile surveillance systems

2.5.1 Sensor and actor networks

Sensor and actor networks (SANET) can be seen as a new class of sensor networks with added functionality [13]. In particular, actuation devices enable the sensor network to interact with the environment in two ways: the environment can be actively sensed (such as PTZ camera) and actuators can manipulate the environment accordingly. The mobile robot is a typical actor that draws the emerging research challenges.

Mobile robots represent a single network entity that performs network-related functionalities systems. At the same time they are able to act on environment using several actuators. Such robots represent resource-rich devices able to move around while communicating and coordinating among themselves and with distributed sensor systems. Therefore, sensor and actor networks are considered by default, providing improved capabilities and

features to solve more sophisticated problems while introducing new challenges such as coordination and communications aspects.

2.5.2 Mobile surveillance systems

Mobile surveillance systems are special cases of SANETs that incorporate self-organized networks of mobile sensing nodes, data and information fusion nodes, acting nodes and control nodes as illustrated in Fig. 2.2. These self-organized nodes can sense collaboratively and continuously the area of interest (AOI) and physically manipulate and interact with it. The sensing nodes represent a set of spatially distributed mobile sensors of different modalities that can sense collaboratively and continuously an area of interest [38, 55]. These nodes can include but are not limited to vision system, sonar/infrared/laser range finders, microphone array or RFID mounted on mobile bases in order to overcome the limitations of the static sensors.

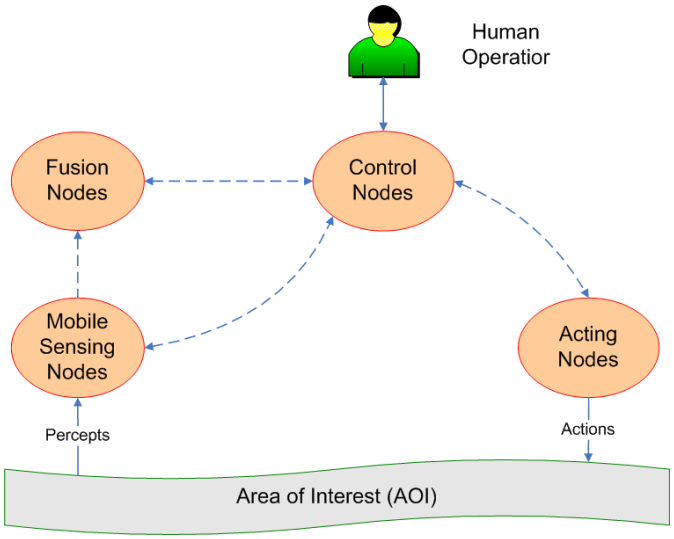

Figure 2.2: Mobile surveillance system.

The functions of each node are given:

- Mobile sensing nodes: are capable of sensing, processing, mobilization and communication with other nodes. They can sample the environment at different locations, exchange the information with other nodes, and collaboratively achieve the required mission.
- Fusion nodes: can be data fusion or information fusion nodes. Data fusion nodes combine sensory data from different sources to generate a more accurate and more

reliable estimate. Information fusion nodes combine information from the sensing nodes about the targets and events in order to determine whether they are behaving normally or if there is any deviation from their expected behavior.

- Acting nodes: may be a direct physical action upon the process, such as moving a camera to keep track an agile target; or a physical making of an electrical circuit, which in turn has a direct effect upon the process. An example would be an actuator (relay) that activates an alarm, a fire extinguisher or hard-kill/soft-kill weapons.

- Control nodes: manages sensing, fusion and acting nodes to provide timely detection, localization, recognition and identification of targets and events, their relationships, activities, and plans, in a given AOI.

2.5.3 Mobile sensing nodes

In a mobile surveillance system, a mobile sensing node is a sensor mounted in a mobile base as mentioned previously. Sensors that are used to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, adding mobile ability are called mobile sensors. Mobile robots are the platform for the function of sensor's mobility.

The movement of each sensor is characterized by its speed and direction, which has two parameters: line velocity v and angel speed a.

Whether a sensor actually detects any given target depends on the relative geometry position of the sensor and the target, physical characteristics of the detection sensor (or fusion of sensors), and the exposure time of the target under sensing scope. The exposure time rule is the longer time of an area under exposure, the greater the sensing ability [36]. When having enough processing time of detection of each sensor node due to limited movement speed and great processing power of nowadays advancement of hardware, the limitation of exposure time can be omitted.

As a rule, detection is more likely the closer the sensor is to the target [18]. Fig.2.3 plots the probability of detection versus the sensor's distance to the target.

In general, the values should fall toward to zero with increasing distance. At same time, the probability of false alarm increases with increasing sensor-target distance.

It is common to approximate a sensor's detection curve in order to facilitate analytical modeling of related searching algorithms. The simplest approximation is depicted in Fig.2.4, definite range law: targets that comes within a certain distance range of the detection sensor are always detected. And targets which are far from that range threshold are never detected. The false alarm of targets is not considered as simplicity. This abstraction

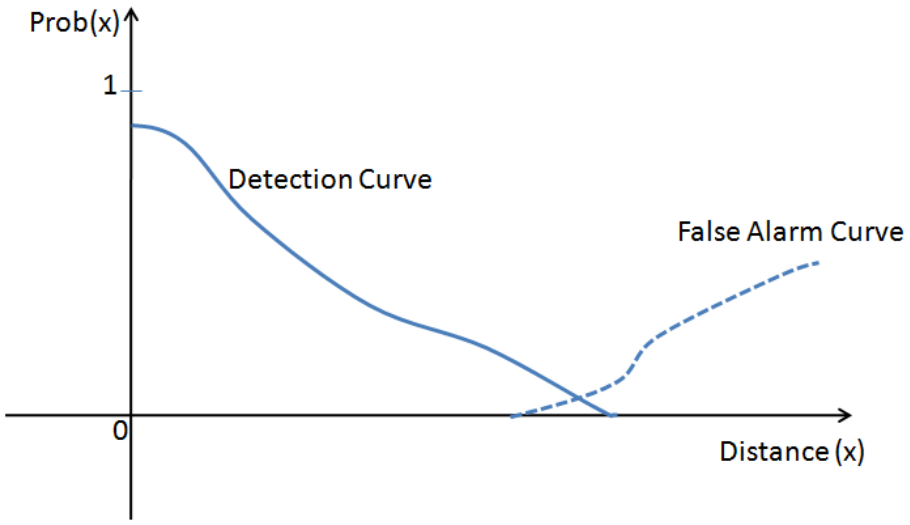

Figure 2.3: Detection probability curve versus sensor's distance to the target.

of sensor thus yields the correct number of targets detected by any single sensor making a single pass through. The sensor is characterized by a single parameter: maximum detection range.

For practical modeling real imperfect sensors, there are several effects needed to take into account, noise, background environment, and target characters. As using more sophisticated target sensing model, the collecting results reflect more realistic performance of system. But it requires extra system resources to model this sensing behaviour in a real-time manner.

2.6 Concluding remarks

Among nowadays surveillance system, the deployment of large amount of static sensors for continuously monitoring the area of interest is effectively in use. But due to the static sensor's limited range and limited sensor types, adding mobile sensors is considered enriching and improving surveillance system capabilities and features. This new direction makes use of mobile more advanced sensors and even takes further action according situation awareness strategy.

Mobile sensor networks started to attract the attention of many researchers as an enabling technology for intelligent mobile surveillance systems [17, 18, 66]. The findings of the research activities helped in tackling and understanding some of the challenging problems of mobile surveillance systems that are still open. These research problems include,

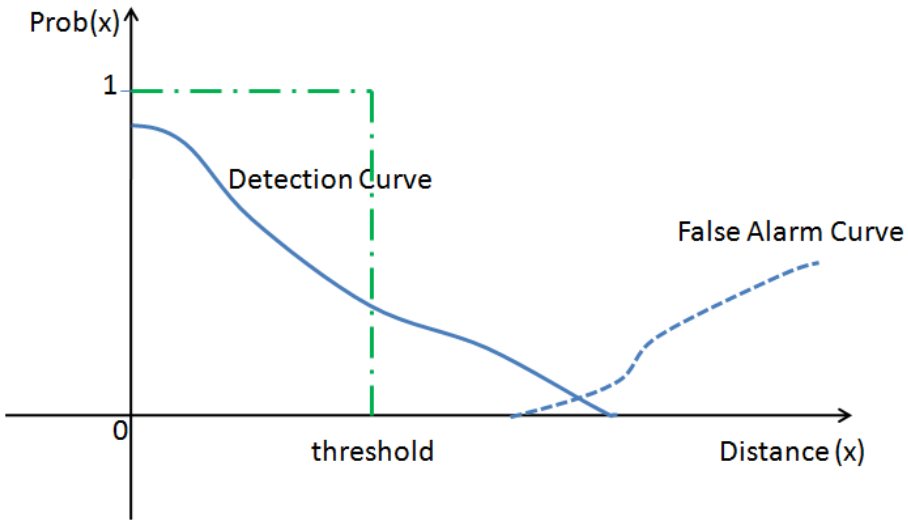

Figure 2.4: Approximation of detection: definite range law.

but are not limited to, mobile sensor deployment, task allocation, sensor management, cooperative object detection and tracking, decentralized data fusion and interoperability and accessibility of system nodes.

This work, as a research topic of MUSES_SECRET project, will focus on the coordination problem of multiple mobile sensors. Next chapter is going to investigate three categories of algorithms for mobile sensor's mobility controlling.

Chapter 3

Mobility Models of Mobile Sensors

Mobile surveillance systems include a vast array of mobile sensing nodes with varying sensing modalities that can sense continuously the area of interest (AOI). These distributed nodes are capable of sensing, processing, mobilization and communication with other nodes. Adding mobility to sensors make them able to sample the AOI at different locations. As a result, locations covered by sensors are dynamically changed, and the area being covered growing larger over time.

Comparing with surveillance system utilizing stationary sensors, performance of mobile sensor network depends not only on the initial network configurations, but also on the mobility models of the mobile sensors [27, 66]. However, adding the dimension of mobility of sensor network for surveillance raises some new challenges such as how to organize the mobility strategy that can maximum system effectiveness?, how are these algorithms robust and adaptable in dynamic environments?

The mobility control strategy for mobile sensors is how to coordinate these distributed nodes in such a way that they can move together in concert that achieves a certain objectives. This objective can be, but is not limited to, maximizing the area and/or target coverage [9, 21], improving secure communications between the mobile nodes (maximizing the radio coverage) [33, 3, 22, 47] or improving the detection rate over agile targets [25].

In this chapter, three categories of mobility models, namely, fully coordinated, fully randomized, and emergent control strategy will be discussed. A novel anti-flocking algorithm to organize multi-mobile sensors searching in a surveillance area is introduced and exploited.

3.1 Fully Coordinated Motion Control

Fully coordinated motion control [17, 18, 38] of mobile sensors is a strategy that comes with all available elements to execute a perfectly coordinated motion and searching pattern. The strategy is comprised by task planning, task assignment and intentionally control the movement of sensing nodes.

In the case of only one mobile sensor, we can follow a "lawn-mower" scanning pattern. When there are a number of mobile sensors, we may choose either divide-and-conquer or group formation strategy. Divide-and-conquer approach divides the monitoring region into N equal fractions and assigns each sensor to scan that area. In the second approach, all the mobile sensors form a tightly sweeping group formation. The whole team is under control to scan the surveillance region.

3.1.1 Divide-and-Conquer

A divide-and-conquer algorithm works by breaking down a problem into two or more sub-problems and solving them in a parallel way [7]. For multiple autonomous mobile sensors, it is a natural routine to adopt the divide-and-conquer algorithm. After task planning and assignment in advance, each element can execute surveillance task (monitoring AOI, target detection etc.) in parallel. It improves system efficiency by splitting and dispensing workload among available resources. Algorithm 3.1 describes the procedures of the divide-and-conquer surveillance searching algorithm.

Algorithm 3.1 Divide and Conquer Sweep Searching

Require: AOI Map, Number of mobile sensors:N

```
Initialization environment:
Setup specifications of mobile sensors
Setup task map of the AOI
repeat
    Task division:
    Equally divide the AOI into N sub-areas
    Assign each sub-area to a mobile sensor
    Task Execution:
    Each sensor scans the assigned sub-area in parallel way
until Command of Task-End
```

The Fig. 3.1 illustrates that three mobile sensors share the surveillance work of scanning by adopting Divide-and-Conquer strategy, i.e. dividing the AOI into three equal subareas and assigning each mobile sensor scanning the designated subarea.

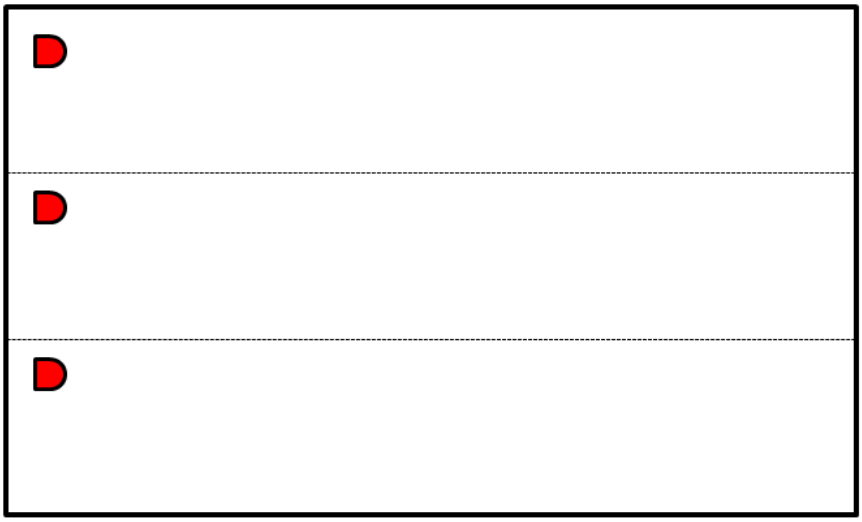

Figure 3.1: Divide-and-Conquer strategy, 3 mobile sensors scan their designated subareas.

There are some obvious drawbacks that prevent the Divide-and-Conquer strategy being widely practiced. First, the region of AOI may not be able to, or very difficult to, divide equally. Then, asymmetry subtasks leads to the efficiency decreasing that acquired from balanced parallel execution.

Second, executing the perfect scanning plan needs employing a very accurate navigation system. Navigational inaccuracies due to sensor and actuator's tolerance leads to systematic gaps in coverage, and missing target detection in the consequence. Fig. 3.2 demonstrates the situation of coverage gaps generating from path tracking varying with planning path. Third, this algorithm is not adaptive to system/environment change and not robust to elements failure. Environmental change, adding or removing nodes will trigger system reconfiguration.

3.1.2 Group Formation

The formation control problem can be summarized as controlling the relative position and orientation of group of sensors while allowing the group to move as a whole body.

From a control point of view, formation control involves two levels of control strategies. One level is the motion control for each individual mobile sensor node to form a suitable formation, such as maintaining their relative position. Another control level is for the whole group, such as transforming from one formation shape to another formation shape, path planning for the center point of the formation should follow.

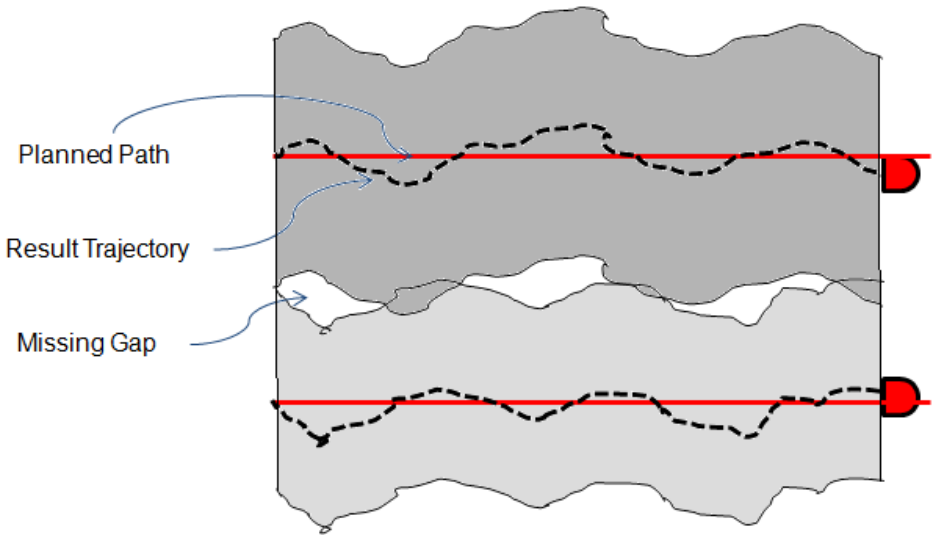

Figure 3.2: Missing gap due to navigational inaccuracies.

- Formation Establishment and Maintaining

  The formation control problem can be summarized as controlling the relative position and orientation of group of sensors while allowing the group to move as a whole body. A well known formation strategy is the leader follower approach [30, 1]. Within a formation of a group of mobile sensors, one is designated as the group leader. The other nodes called followers maintain a desired distance and orientation relative to the leader. Thus, if we ask the leader to move along a predefined trajectory, the whole group will move as a whole following the same routine. The multi-mobile sensors mobility control is simplified to control the movement of the leader and maintain the structure of the formation at the same time.

  The chosen strategy of leadership can be varied. A good candidate is choosing the one that is the nearest to the target as the leader. The method can produce a variation of floating leader formation control strategy, which the leadership is chosen timely according to its present position nearing the target, while tracking the moving target. After the leader is decided, followers have two basic strategies to maintain structure stability: $l-\phi$ and $l-l$. As shown in Fig.3.3(a), the relative position of the follower is achieved through maintain a desired distance l and a desired relative bearing ϕ with the leader. The Fig.3.3(b) shows the scenario of $l-l$ control strategy. It is designed to maintain desired distances $l1$ and $l2$ with two referrer nodes.

  In practice, a multi-sensor formation control involves both the $l-\phi$ and $l-l$ control strategies. For triangle style formation, the Fig. 3.4 illustrates that the mobile sensors in both wings use the $l-\phi$ strategy while the sensor nodes in the middle use

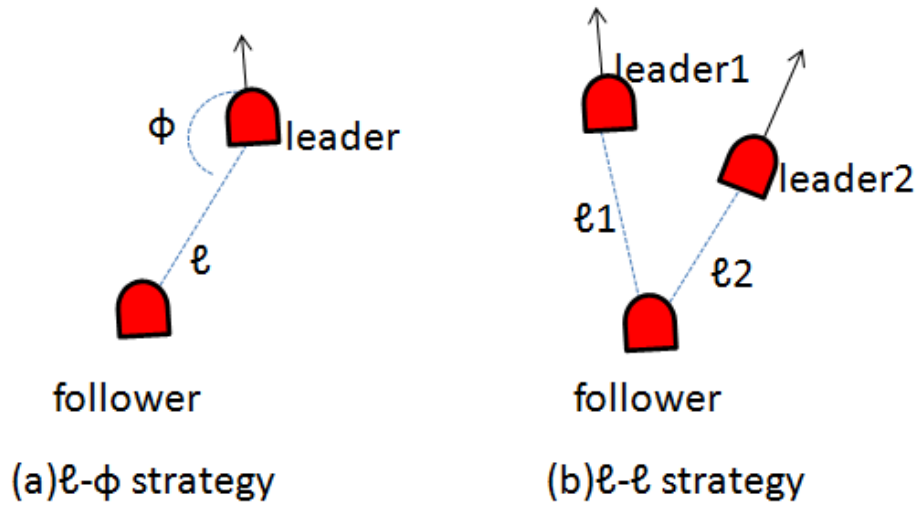

Figure 3.3: Two basic control strategies for maintaining a formation shape.

the $l - l$ strategy. However, another basic group formation, the line style that the mobile sensor nodes travel as a queue, uses the $l - \phi$ strategy only.

- Formation Transform

  Let's consider the group of mobile sensors forming a triangle shape in an free of obstacles open area. It can move in a stable way while maintaining the formation shape until it encounters such scenarios as big obstacles, going through a narrow gap, going through a long zigzag corridor etc. When the leader arrives at these scenarios, it is necessary to transform the formation into a passable shape to avoid collision and pass it. Fig.3.5 explains the system architecture of formation control with ability of obstacle avoidance through formation transforming.

  In the situation of encountering obstacles forming a narrow gap, as the Fig.3.6 illustrates, that can be passed by one mobile sensor a time, and the strategy is to transform the triangle shape into a line style. It can be assigned the relative position of each follower by the leader according predefined topology. And, after the whole formation passes through the narrow gap, the leader can order to change back to triangle shape again.

- Group Formation Sweep Searching Algorithm

  Based on the group formation strategy discussed above, multi-mobile sensors sweep searching in group formation approach is explained in the following Algorithm 3.2.

- Comments on group formation control

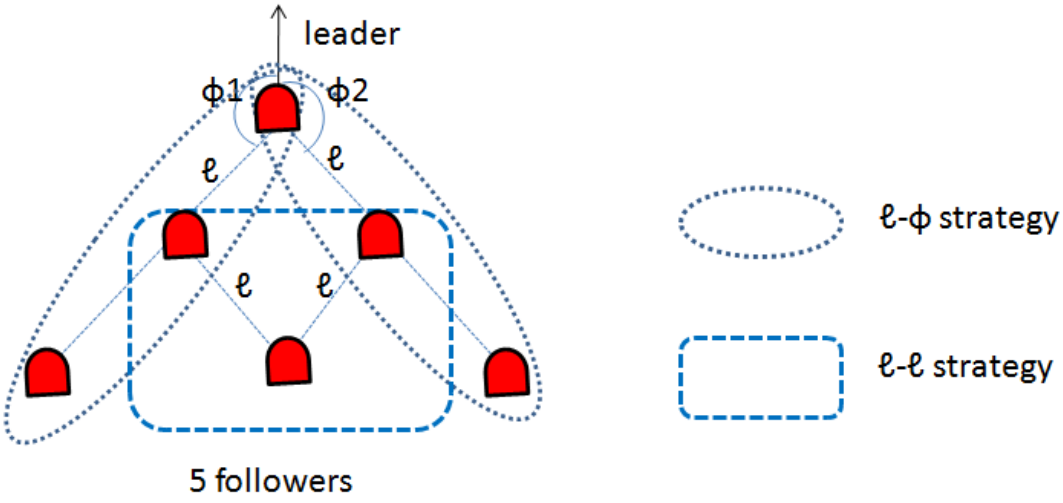

Figure 3.4: Formation structure compound with two control strategies.

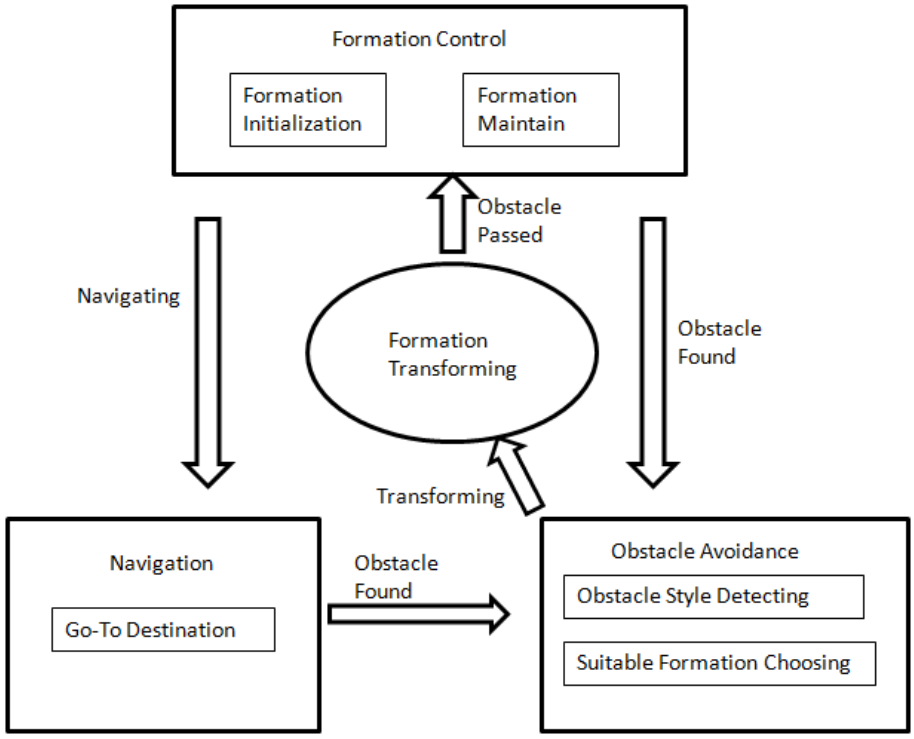

Figure 3.5: Architecture of formation control with obstacle avoidance (from Liu et al. [30]).

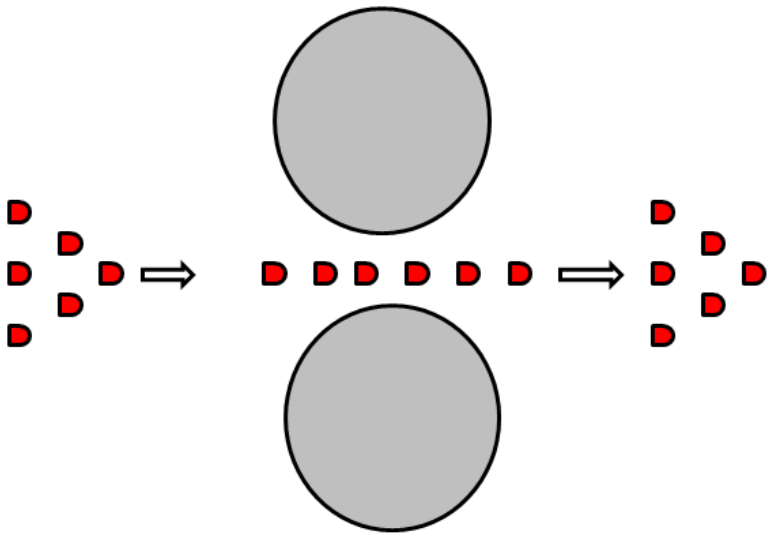

Figure 3.6: Formation transforms between triangle and line style in order to pass a narrow gap.

Algorithm 3.2 Group Formation Sweep Searching

Require: AOI Map, Number of mobile sensors:N

Initialization environment:
Setup specifications of mobile sensors
Setup task map of the AOI
Team formation of N mobile sensors into a line shape
Maintain leader-follower formation through controlling relative distance and/or angle
repeat
    Path planning for the team to scan
    Send sweeping execution plan to leader, followers maintain formation shape
    if Obstacle Found **then**
        Leader decides a suitable formation shape
        Leader sends followers command of formation transform
        if Obstacle Passed **then**
            Leader sends followers command of transforming back
        end if
    end if
until Command of Task-End

As we can see from the above description, the intentional leader-follower control strategy for a group of mobile sensors movement is practical and easy implemented. The main criticism of the leader follower strategy is that the formation's robustness mainly depends on the leader unit. Any faults of the leader leads to group malfunction.

Another deficiency is that the control system is based on the full communication between the mobile sensors and each sensor's unique identification. Therefore, for n mobile sensors in a formation, the cost of communication is $O(n^2)$. The communication cost is too high when the number of mobile sensors increases. Furthermore, because the mobile sensors are working in a dynamic environment, the communication range and bandwidth usually are limited and not in high quality.

From the observation of the experiments of [30], applying this leader follower strategy, any leader's disturbance affects the follower's motion and the influence will be passed and amplified gradually to the last follower. This situation is even worse in the line style formation strategy. This can be partially amended by loosen the control parameters tolerance.

Additionally, in order to keep the formation in a steady shape, the minimal rigidity of formation is needed. The theoretical analysis of operations on formation's splitting, merging, and closing ranks is discussed in [1].

In fact, the fully coordinated motion control is a superior strategy because of its careful task planning and assignment. However, real world constraints, such as navigation inaccuracies and unreliable communication, affect the performance of this strategy and limit its ability to achieve full coverage. Further, when adopting the coordinated strategy, a patterned search path may permit agile targets to evade capture.

3.2 Fully Random Motion Model

In the fully random search model [17, 18, 27], number of mobile sensors are wandering completely randomly, with only these capabilities: staying within the designated search area and avoiding collision with each other and any unexpected obstacles.

Formally, this mobility model is defined as moving velocity is limited with maximum speed V_s, and with moving direction $\theta \in [0, 2\pi)$ according to some distribution with probability density function $f(\theta)$. Liu et al. proved in [27] that the random searching strategy also approaches to cover all area as time goes to infinity. The covered area increases over time depends on the speed of mobile sensors and their static sensing range radius.

Algorithm 3.3 shows the procedures of fully random searching.

Algorithm 3.3 Fully Random Searching

```
1: Initialization environment:
2: Setup specifications of mobile sensors: default line speed v
3: repeat
4:    Sensing Obstacles
5:    if obstacles exist then
6:       Invoke obstacle avoidance routine
7:    else
8:       Invoke random function with return value that uniformly distributes ∈ [0, 2π)
9:       Move to the direction according to the return value of random function
10:   end if
11: until Command of Task-End
```

There are three main advantages of this strategy. First, the random mobility model is easy to implement resulting in applications that deploy a large number of low cost simple mobile sensors. The system level mission-specific functions and performance requirements can be achieved through high nodes redundancy. Second, the random mobility model inherently adapts to dynamic environment where prior knowledge of the region of interest is not available. The third advantage is that because mobile sensor's path is not predictable, so that a mobile target cannot make use of its observations or knowledge to predict its path and thereby evade detection.

However, the inevitable shortcoming is that the random search mobility strategy is clearly not an efficient one because of repeating searching of each sensor and among sensors. On the other hand, there is no guarantee of covering all area with a time frame, although it is proved that whole region will be covered as time goes to infinity when adopting this random mobility strategy.

3.3 Emergent Motion Control Strategy

Self-organizing formation is an emergent process of making whole forms by local interactions of distributed simple autonomous elements without global information at all and without depending on the initial position and orientation of the elements. These simple autonomous elements store local information and guiding rules needed for the colonial self-organization. Additional information is cooperatively generated as the organization proceeds following external stimuli. The outcome is an adaptable complex system that can perform many tasks, learn and change itself accordingly. The main properties of self-organizing systems are the absence of central control, emerging structures, resulting complexity and high scalability [13].

3.3.1 Flocking Algorithm

In the world of living things, many astonishing examples of self-organization can be easily noticed. For example, fish schooling is a form of self-organization that has advantages in the energy consumption as one fish can utilize the pressure field created by the next fish when they cruise in a close group. Bird flocking is also a self-organization behavior controlled by three simple rules: Separation, Alignment and Cohesion [50]. The objective of the first rule is to avoid collisions with nearby flock-mates. Alignment or velocity matching means try to match velocity with nearby flock-mates and cohesion or flock centering aims to stay close to nearby flock-mates. The three heuristic flocking rules are illustrated in Fig.3.7.

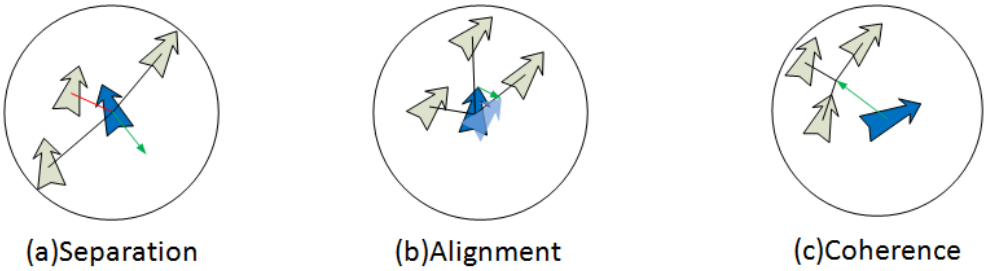

Figure 3.7: Three basic flocking rules.

Applying these three rules flocking algorithm is called no navigational feedback flocking. The criticism of this algorithm is that it fails to form one structure flocking and leads to regular fragmentation for generic initial states and large number of flocking members. The fragmentation phenomenon is arisen from two aspects:

- Inherent property of flocking: the limited eyesight of flock members.
- No overall system objective.

Under the condition that system objective is free space searching and rescuing, this fragmentation is helpful to balance increasing searching visible range and reducing redundancy searching comparing with only one single flock structure.

Olfati-Saber gives the flocking algorithm considering these three rules in an open space using the concept of control the second derivation of mobile sensors position, motion force [45, 46].

The Fig.3.8 shows the effects when adding target tracking feedback with three basic flocking rules. The three sub-forces are explained as:

- f1: force to control flock-mates follow rules of Separation and Coherence ;

- f2: force to control flock-mates follow rule of Alignment;
- f3: force arisen from the system objective–target tracking.

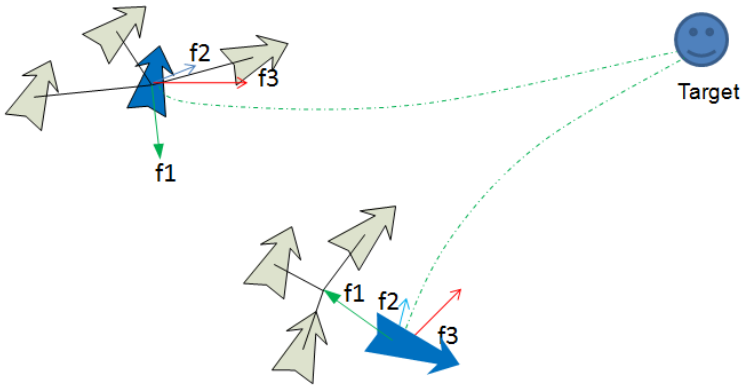

Figure 3.8: Adding target tracking feedback to three flocking rules.

However, the flocking algorithm results in massive mobile sensors assemble around a predefined target. This is proved to improve the performance of target tracking, which is a different objective comparing to this research that tries to cover more area and searching and detecting targets.

3.3.2 Diffuse Reflection Algorithm

Inspired by the idea of light reflecting from a matte surface, diffuse reflection algorithm is a strategy that searching unit travels as far as possible following a straight line path until reaching boundaries and changing directions like light diffuse reflection [34, 18]. McNish found that the diffuse reflection algorithm reliably provided uniform search coverage.

- Diffuse Reflection in a circular search area

  The diffuse reflection algorithm specifies a random distribution of reflection angle θ with probability defined as:

$$Prob(\theta) = constant \times \sin(\theta) \tag{3.1}$$

where θ measures the angle of the chord from the tangent to the boundary at the reflection point. Lalley and Robbins mathematically proved the search coverage is uniformly distributed over a circular disk while adopting the diffuse reflection strategy [24]. Fig.3.9 demonstrates the circular situation.

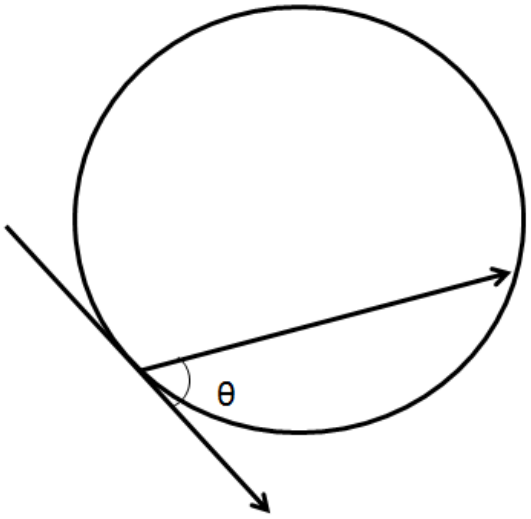

Figure 3.9: Diffuse Reflection in circular search area.

- Generalization to convex area

  The above diffuse reflection has a very strict constraint, working only in circular search area, which limits the scope of realistic applications. In [18], author presented a generalization algorithm that works in convex search area. The reflection angle follows the distribution as:

$$Prob(\theta) = constant \times K(\theta) \tag{3.2}$$

  where $K(\theta)$ is the length from the chord to the opposite boundary. The Fig.3.10 explains the $K(\theta)$ generalization diffuse reflection strategy.

  The diffuse reflection algorithm is optimal in that it leads to uniform searching coverage over the circular or convex area on a per-chord basis. However, to implement this algorithm, the mobile sensor must be able to determine its position and the distance to the opposite boundary in all directions. It is not practical in most real world applications.

  Furthermore, the algorithm is working basis on the single unit. When system owns multiple mobile sensors, there need adding other rules for the purpose of coordination among sensors, such as divide-and-conquer.

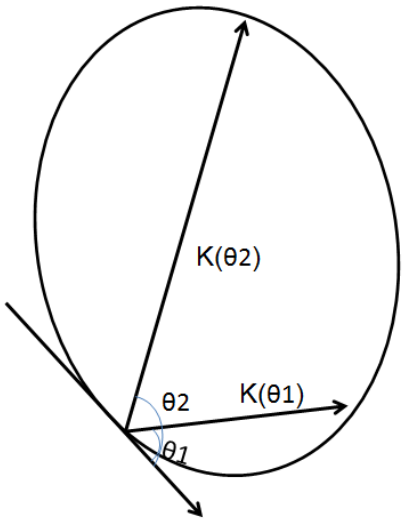

Figure 3.10: $K(\theta)$ generalization diffuse reflection in convex search area.

3.4 Anti-Flocking Algorithm

3.4.1 The Social Behaviour of Solitary Animals

Unlike many social animals that stay together as a group showing collective behaviours such as flocking, there are some animals that survive thanks to different social behaviours called solitary behaviours. Animals like tigers, spiders are examples of these kinds of solitary animals, which try to set apart from each other in their daily life activities of foraging, except mating or caring for newborn offspring [19, 23, 56].

The solitary animals behaving like this show overall beneficial to all species members to maximum searching coverage without overlapping that are implemented by visible finding neighbourhood and stigmergic communication or communication via environment, such as marking trees by spraying of urine and anal gland secretions, as well as marking trails with scat. So, the collective intelligence among solitary animals can also be thought as high-level cooperation that is based on different conditions of flocking animals. Here it is called Anti-Flocking behaviour.

3.4.2 Anti-Flocking Algorithm

Rules of Anti-flocking algorithm can be summarized as follows:

1. Collision avoidance: stay away from the nearest obstacle that is within safe distance;

2. De-centering: attempt to move apart from its neighbours;

3. Selfishness: if neither the above two situations happens, move to a direction which can maximize one's own interests.

As illustrated in Fig.3.11, the first rule is easy to understand and implement. Mobile sensor moves away from its nearest obstacle when being detected through its range finders, whatever it is an outer obstacle or a neighbour unit. The threshold of safe distance can be achieved through experiments and other fuzzy or soft computing models.

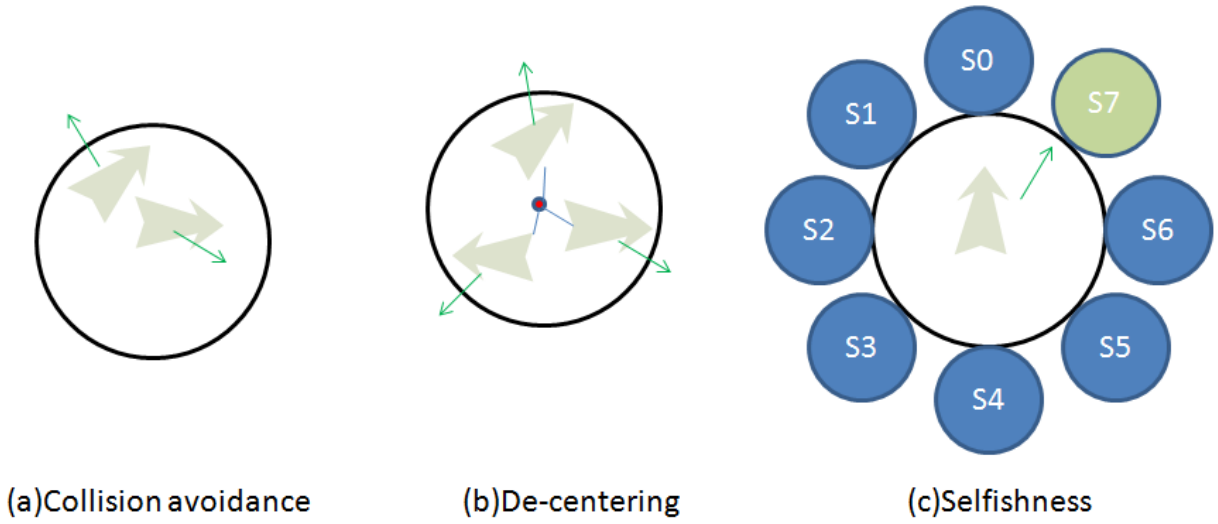

Figure 3.11: Three rules of Anti-Flocking algorithm.

The second rule is a de-centralizing method that scatters available sensing resources around the whole volume of interested monitoring area. With a setting of neighbourhood radius, which is usually decided by the communication range of mobile sensors, each mobile sensor moves away from the center of its neighbours.

The third rule, selfishness, is applied when neither obstacles nor neighbours appears. In this case, the mobile sensor moves to the direction that maximizes the sensor's own interest. As shown in Fig.3.11(c), the mobile sensor considers all its eight surrounding possible candidate directions and chooses moving to the direction $S7$ which has maximum gain according to some reward calculation.

In the surveillance application, the gain of a mobile sensor node is the area that has the high possibility of events existence. Assuming that events happen following uniform distribution in the AOI, and also are uniform distributed along time slot, this can be represented as a memory-less Poisson counting process [14]. The possibility of events

appearing in an area until time t passed follows this distribution:

$$Pr(N(t) \geq 1) = \begin{cases} 0 & \text{for } t < 0 \\ 1 - \lambda_0 e^{-\lambda * t} & \text{for } t \geq 0 \end{cases} \tag{3.3}$$

Where $N(t)$ is the number of appearing events till time instant t, λ_0 is the initial possibility at time $t = 0$, and λ denotes the appearing rate of events. So, it can be concluded that the longer time an area is not visited, the higher possible events exist in that area. So, we can set the gain of a mobile sensor optimal moving direction to be the area that has the longest time not being visited.

Algorithm 3.4 describes the proposed anti-flocking searching strategy.

Algorithm 3.4 Anti-flocking searching

Require: AOI Map, Neighborhood Radius:r_n, Gain Growth Rate:λ

```
Initialization environment:
Setup specifications of mobile sensors
Setup gain model of AOI
repeat
    Sensing obstacles
    Sensing neighbors
    if obstacles exist then
        Invoke obstacle avoidance routine
    else if neighbors ≠ null then
        Calculate centroid of neighbors
        Move to the opposite direction to the centroid
    else
        Calculate gains of surrounding directions
        Move to the direction with maximum gain
    end if
    Update gain of AOI
until Command of Task-End
```

3.5 Summary

In this chapter, three categories of mobility models of mobile sensors are described: fully coordinated model, fully random model, and emergent motion control model.

In the first section, two algorithms that belong to coordinated motion control are discussed, i.e. divide-and-conquer and group formation. Following, the fully random model

is formalized. Then, the section of emergent control strategy discusses artificial systems that can analog the behavioural mechanisms inspired by nature. Flocking algorithm and diffuse reflection algorithm are two important examples of self-organized approaches for motion and search. Further, a novel anti-flocking algorithm that mimics solitary animal's behaviour is proposed as a strategy to organize multiple mobile sensors mobility.

The Table 3.1 summarizes the characters of discussed mobility models of multiple mobile sensors.

Mobility Model	**Pros**	**Cons**	**Algorithm**
Fully Coordinated	High efficiency, good controllability	Non-scalable, non-robustness	Divide-and-Conquer [7]; Group Formation [30]
Fully Random	Easy implementation, environment adaptability	Low efficiency, unpredictable performance	Fully Random [27]
Emergent Control	High scalability, flexibility, robustness, environment adaptability	difficult global controllability, unforeseen system status	Flocking [50]; Diffuse Reflection [34, 18]; Anti-Flocking

Table 3.1: Mobility models of mobile sensors.

Chapter 4

Evaluation Metrics

In the community of research and developing practical surveillance system to improve public security and environmental monitoring, there is a fundamental question that needs to be tackled first: How to evaluate the effectiveness of surveillance system? From the viewpoint of end users, answering this question can also lead to the answer of: How well a region of interest is monitored using the surveillance system?

Effectively evaluating the performance of surveillance system is an important step to examine, analyze, highlight diverse aspects on system weakness and help for further improvements.

The measurement of effectiveness (MOE) of a surveillance system can be gathered through the following two approaches. One approach is with objective related to deployment coverage [35, 9, 27]. Another approach is with the objective related to picture compilation [4, 41, 28]. The first part of this chapter will introduce the general taxonomy of evaluation metrics in the literature of assessing performance of surveillance system.

Because this research involves mobile sensor and its mobility characteristics, the dynamic features are also important aspects to be considered. In the later part of this chapter, instantaneous area coverage, cumulative area coverage, detection rate, average detection time and maximum detection time will be described in detail as the system evaluation metrics used in the scope of this research.

4.1 General Taxonomy of Performance Evaluation in Surveillance System

A general picture of surveillance system performance evaluation is comprised with the two branches related to the scope of this thesis: measuring deployment coverage and measuring

picture compilation. Fig.4.1 lists the general taxonomy that reflects different aspect to address the evaluation problem of a surveillance system.

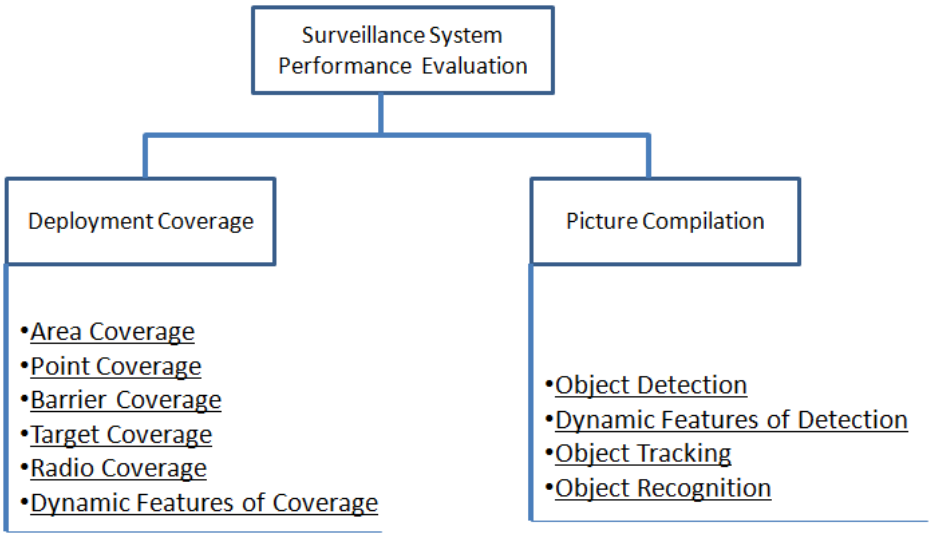

Figure 4.1: General taxonomy of surveillance system performance evaluation.

In intelligent surveillance system, one important system objective is to find optimal sensor deployment methodology that maximizes coverage of the Area of Interest (AOI) or Object of Interest (OOI). Assessment of this system function leads to measure the following aspects: 1). Spatial Coverage: includes area coverage, point coverage, barrier coverage, and target coverage; 2). Radio Coverage: addresses problem about communication, connection and signal monitoring.

Another system objective of surveillance system is generating a common picture where all the information provided by multiple sources is represented in a common workspace, assisting decision makers to understand the situation [4]. This process of picture compilation involves the following sub-processes: 1). Object Detection: includes employment sensing and processing power in the AOI in order to determine the presence or absence of objects or information related to objects; 2). Object Tracking: includes the employment of sensing and processing power to determine the localization and features related to movement of objects; and 3). Object Recognition: leads to high level conclusion about characteristics of objects, classification the types of objects.

4.1.1 Measuring Deployment Coverage

The coverage concept [35, 9, 27] is a measure of the quality of services of the sensing function and is subject to a wide range of interpretations due to different kind of applications.

Concerning different focus of surveillance system, the spatial coverage problem is divided into: area coverage, point coverage, barrier coverage, and target coverage. When introducing mobile sensors into the system, dynamic features of instantaneous coverage and cumulative coverage are additional dimensions of MOE.

- Area Coverage

  Area Coverage is the fraction of AOI that is covered by monitoring sensors. Fig.4.2 shows an example of a deployment of 3 sensors, each with a radius range of covered region which is decided by sensor's static specification. The sensor network area coverage is the union of all sensors coverage, i.e. the total shadowed area in the Fig.4.2

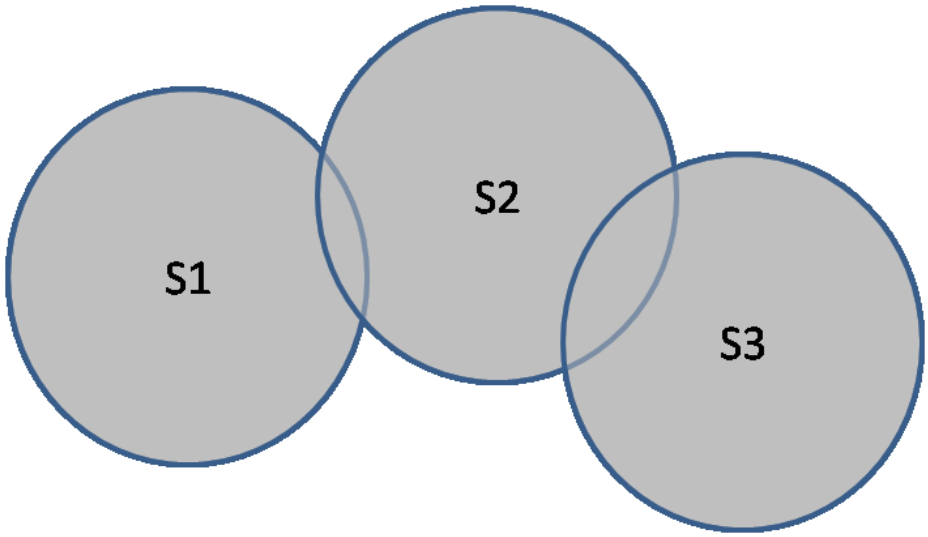

Figure 4.2: Area coverage of 3 sensors.

Concerning area coverage problem, the main objective of surveillance system is to cover/monitor an AOI. It works based on the assumption: the larger part of interested area being monitored, the better the surveillance system works.

Obviously, area coverage is decided by sensor's density, sensing range and its deployment. With fixed number of sensors and predefined sensing range, the sensors' deployment plays a critical role in improving area coverage. Overlapping with each other is lowering the usage efficiency of the whole surveillance sensor system.

The research in [9] exploited another aspect that is to schedule the activeness of redundant sensor nodes to improve system energy efficient while maintain required area coverage.

Besides sensor deployment to maximize area coverage, another important constraint in sensor network is connectivity. A network is connected if any active sensor node can communicate with other active nodes, directly or through ad hoc relays. Once the sensors are deployed, they cover the AOI and also have constraint of being connected so that information collected by sensors can be sent back to controllers. So, surveillance system objective is to maximum sensing area coverage while maintaining system connectivity.

- Point Coverage

  In the point coverage problem, the objective is to cover a set of points (critical positions given by external sources). Fig.4.3 demonstrates an example of 3 sensors deployed to cover all suspected points (marked with stars).

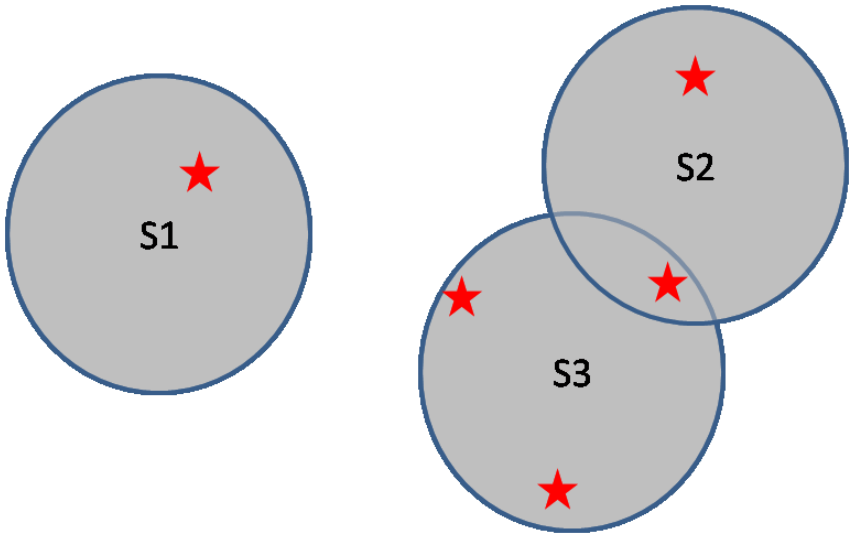

Figure 4.3: Point coverage, stars are points to be monitored.

The point coverage scenario addressed in [8] has military applicability. It considers a limited number of known locations that need to be monitored. As a result, the objective of surveillance system alternates to cover these discrete points in the AOI, through sensors configuration and deployment.

For the problem of point coverage, scheduling mechanisms for energy efficiency and maintaining network connectivity are also needed to be taken into account.

- Barrier Coverage

  From the concept given in [17], the barrier coverage is to achieve a sensor arrangement with the objective of minimizing the probability of undetected targets penetration through a predefined barrier.

Fig.4.4 shows a barrier coverage problem with a barrier path to be monitored. The selection of the monitored barrier path (green line) depends on application's requirements.

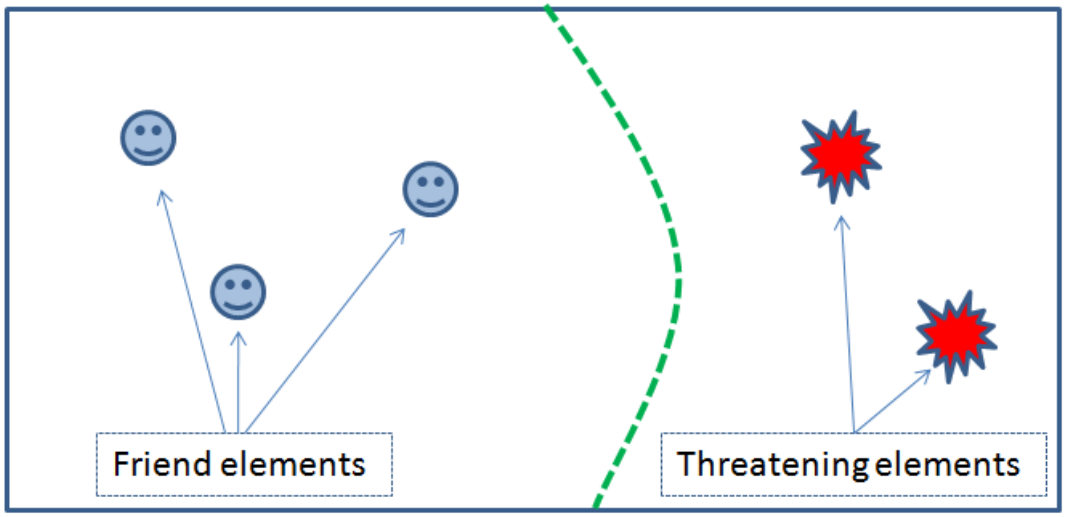

Figure 4.4: Barrier coverage (green dotted line) to protect "friend elements" from intrusion of "threatening elements".

A barrier coverage model proposed in [35]: given a field instrumented with sensors and the initial and final locations of a mobile agent to move through the field, determines a maximal breach path (MBP) and the maximal support path (MSP). The MBP corresponds to the worst case of barrier coverage and has the property that for any point on the path, the distance to the nearest sensor is maximized. Meanwhile, the MSP represents the best case of barrier coverage that the distance to the closest sensor along the path is minimized.

- Target Coverage

  Target coverage is finding an optimal viewpoint, or a group of viewpoints (case of multisensory) that satisfy task requirements to inspect/track a target with better visibility. Authors of [21] described a multi-camera system that autonomous positioning to improve target coverage. Concerning a 3-D polyhedrons scene, the proposed system uses the metric of the percentage surface area of the interested target that is covered across mutli-sensors' viewpoints.

  The Target coverage is an important measure in target inspection, tracking and further leading to classification.

- Radio Coverage

  Radio coverage in sensor network system focuses on signal covering problem to keep sensor's connectivity while maintaining sensing coverage [64]. Usually radio communication has a larger area than sensor's effective range.

- Dynamic Coverage Problem with Mobile Sensors

  When introducing mobile sensors into a surveillance system, it adds different view angels of performance assessment because the system demonstrates dynamisms as sensors own ability of moving. An area that was not covered by sensors at a time instant can be covered as time going and a mobile sensor passing by that area.

  In the literature of mobile sensor networks [27, 48], considering a new time dimension, two new aspects of area coverage are exploited: instantaneous area coverage, and cumulative area coverage.

4.1.2 Measuring Picture Compilation

When sensing model of each sensor is given, injecting targets (object of interest, OOI) is a realistic method to evaluate the performance and working status of a whole surveillance system. Results of target detection can not only assess system performance from an end-user point of view, but also address system characteristics and highlight algorithm weakness for further improvements. The following various metrics help to analyze the diverse aspects of surveillance system [41, 28].

- Metrics for object detection

  In the field of object detection and in the case of imperfect sensors, the performance is evaluated using these measures: Precision, Sensitivity, and F-score [41, 14].

  In order to explain the definition of these measurements, consider a cross contingency table, Table 4.1, showing a detection result of two categories (positive and negative category). TP is the number of OOI that are the correctly detected. FN is the number of OOI that are missed. FP is the number of false alarm. TN is the number of objects that are not belong to OOI and are correctly disengaged.

GroundTruth	Positive	Negative
Detection positive	TP (True-Positive)	FP (False-Positive)
Detection negative	FN (False-Negative)	TN (True-Negative)

Table 4.1: Contingency table of object detection result.

Formula 4.1 and 4.2 give the definition of Precision and Sensitivity according to the terms in Table 4.1. The Precision is defined as the percentage of the number of correctly detected positive objects comparing the total number of detections ($TP+FN$), while Sensitivity is defined as the percentage of the number of correctly detection positive comparing the number of all ground truth positive objects ($TP+FN$). The F_{score} measure is a harmonic mean of the Precision and Sensitivity defined in Formula 4.3.

$$Precision = TP/(TP + FP) \tag{4.1}$$

$$Sensitivity = TP/(TP + FN) \tag{4.2}$$

$$F_{score} = (2 * Precision * Sensitivity)/(Precision + Sensitivity) \tag{4.3}$$

- Metrics for object tracking

  The metrics "2D/3D distance" measure the average of the 2D/3D distance between gravity centers of detected objects and corresponding reference objects. These metrics help to determine the detection precision of object localization.

  For the tracking task, there are one main metric, "tracking time", and two other complementary ones, "object ID persistence" and "object ID confusion". The metric "tracking time" measures the percentage of time during which an object of interest (OOI) is detected and tracked. The metric "object ID persistence" computes over the time how many tracked objects are associated to one reference object (ID persistence). In the contrary, the metric "object ID confusion" computes the number of reference object IDs per detected object. An example of confusion is the case of two objects meeting, and the object ID is exchanged.

- Metrics for object recognition

  Object recognition is high level surveillance result conducting from information fusion of objects detection and scenario context. For the object recognition task, a set of classification evaluation metrics are described in [14] when several types of objects which need to be classified, i.e. precision, recall, F-measure, and error rate.

4.1.3 Summary

The following Table 4.2 summarizes the performance evaluation metrics used in the literature of surveillance.

Assessment Objective	Evaluation Metrics	Measurement
Deployment Coverage	Area Coverage	Covered Area of AOI [35]
	Point Coverage	Covered Points of AOI [8]
	Barrier Coverage	MBP, MSP [35]
	Target Coverage	Covered Area of OOI [21]
	Radio Coverage	Signal Covered Area [64]
	Dynamic Features of Coverage	Instantaneous Coverage and Cumulative Coverage [27]
Picture Compilation	Object Detection	Precision, Sensitivity, F-score [41]
	Dynamic Features of Detection	Detection Rate, Detection Time [27]
	Object Tracking	Euclidean distance, Tracking time, Object ID persistence, Object ID confusion [41]
	Object Recognition	Precision, Recall, F-measure, Error Rate [14]

Table 4.2: Performance evaluation metrics for surveillance system.

4.2 Proposed Evaluation Metrics

For the specific of this research, studying the effect of mobility models of mobile sensors, we adopt the evaluation metrics of both area coverage and object detection. Further details are given in the following subsections.

4.2.1 Area Coverage

- Instantaneous Area Coverage:

  At a time instant t, the instantaneous area coverage is the fraction of the geographical area covered by mobile surveillance system versus the whole area of surveillance region, denoted as $Ai(t)$.

  Instantaneous area coverage is union operation (as Fig.4.5 shows) of all sensors coverage at time instant t, and is decided by sensors density, sensing range, and position configuration. Overlapping with each other is obviously lowering the usage efficiency of the whole mobile surveillance system.

  Instantaneous area coverage is important for applications that require simultaneous area coverage. These applications need identify emergency events in critical timely way, such as surveillance system for fire detection. Sensor density requirement and properly deployment are the main focus of this kind of task.

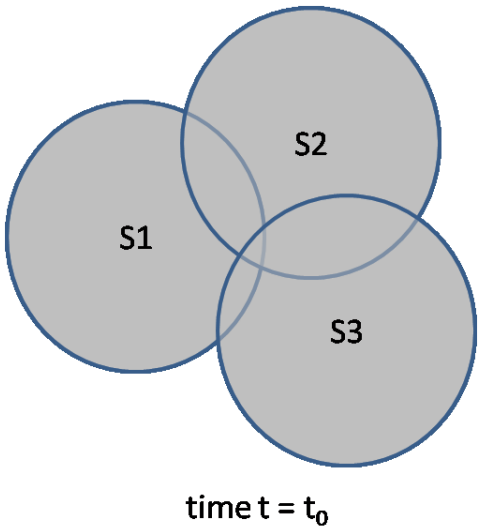

Figure 4.5: Instantaneous area coverage of three mobile sensors at a time instant.

- Cumulative Area Coverage:

  For a time interval $[0, t)$, cumulative area coverage of a mobile surveillance system is the fraction of the geographical area covered by at least one sensor at least one time within the time interval $[0, t)$, denoted as $Ac(t)$. Fig. 4.6. shows a situation of 2 mobile sensors from time t_0 to t_1. The union of the shaded region represents the cumulative area coverage during the time interval $[t_0, t_1)$.

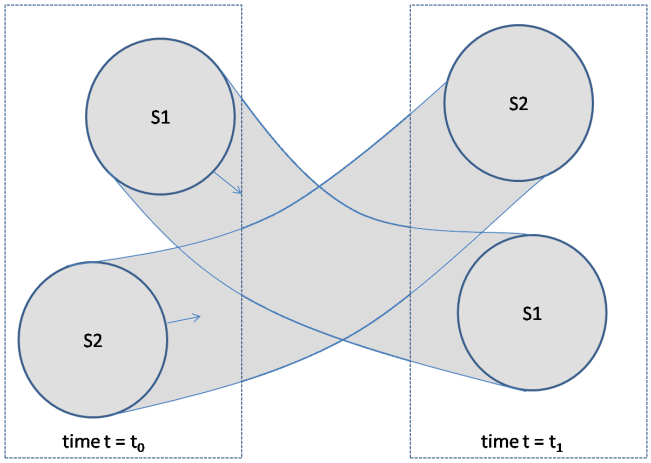

Figure 4.6: Cumulative area coverage during time interval $[t_0, t_1)$.

4.2.2 Object Detection

- Detection Rate:

  Setting a scenario that there are n targets located in the surveillance region, the detection rate is the percentage of targets have been detected after a threshold time t' passed, as expressed in Formula 4.4:

$$DetectionRate = (\#DetectedObject)/(\#ReferenceObject) \tag{4.4}$$

- Detection Time:

  Consider a target appearing in the surveillance region (AOI) at time $t = 0$, the detection time of this target is defined to be the passing time until the target is first detected by any sensor.

  To evaluate detection time of a mobile surveillance system, besides average detection time, maximum detection time is worthy to be reported because it is an important indicator in some applications with need of a guaranteed worst case.

Chapter 5

Experiments and Discussions

This chapter presents conducted experiments of both simulation and real robots platform. Three types of algorithms are implemented on a distributed multi-agent architecture. Area coverage and detection effectiveness results are analyzed. Discussions on system performance, features and limitations of different mobility models are summarized in the last section.

5.1 Experimental Setup

A multi-agent system has been implemented to examine the efficiency of different mobility models of mobile sensors as shown in Fig.5.1. Multi-agent system paradigm introduces a number of new abstractions and design/development issues when compared with more traditional approaches to software development [62]. This paradigm is well-suited for use in applications that involve distributed computation, concurrent processing capabilities or communication between components which is the case in mobile surveillance systems. In this multi-agent system, each agent possesses the ability to behave autonomously to react to external stimuli and the ability to exhibit opportunistic, goal-directed behavior to accomplish a given task individually or in collaboration with other agents.

The implemented multi-agent system encompasses the following agents as shown in Fig.5.1:

- The Control Agent: this agent interfaces with human operator (accepting command, response system statuses); generates task plan; coordinates all related agents to execute the task.

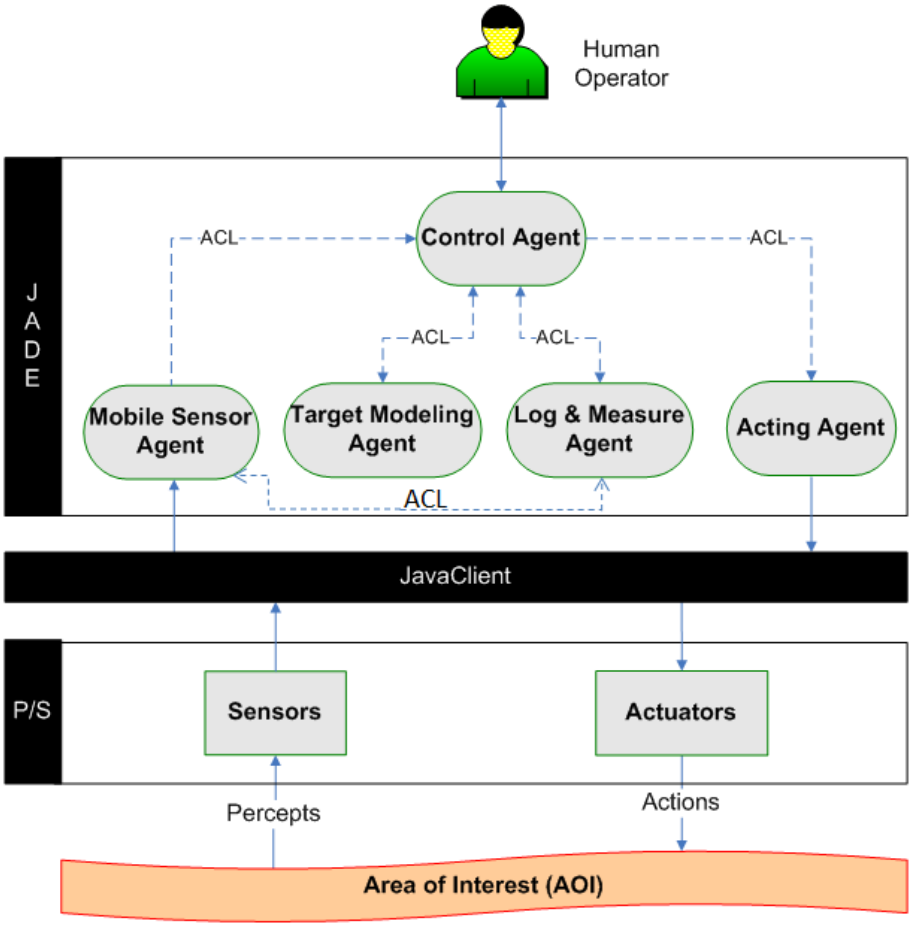

Figure 5.1: Multi-agent system.

- Mobile Sensor Agent: this agent represents the main controller of the mobile sensor to retrieve sensor information and to drive it in the AOI.
- Target Modeling Agent: in our experiment, the target modeling agent generates target events according to a predefined distribution.
- Log and Measurement Agent: this agent is used to collect experimental results and write to log file for further analysis.
- Acting Agent (to be expanded in future work): take intelligent action/serial of actions to physically manipulate and interact with the AOI.

In the following experiments, JADE (Java Agent Development Framework) [16] is used as agent management platform. The mobile sensor's driver and simulation is constructed based on Player/Stage [60]. The Java agent communicates with Player/stage through a JavaClient library [10]. More details about these tools can be found in appendix A.

5.2 Simulation

5.2.1 Simulation environment parameters

By choosing the Stage [60] simulation software package, the experimental environment is set as follows. The area of interest is a rectangle area with dimension of $35meters \times 30meters$. There are three mobile sensing nodes in the system. The mobile sensor uses model of KheperaIII [58] with sonar sensors and infrared sensors. The default line speed is set to $0.5m/s$. Target detection radius $r = 2m$, and neighbourhood radius $Nr = 4m$. Instantaneous area coverage and cumulative area coverage results are based on 5 executions, and maximum running time (threshold) is set to $t' = 1000s$. For event detection, we model 100 static events that uniformly distributed in the scenario. To avoid randomness, 10 runs are executed which result in collecting totally 1000 events detection results. Table5.1 lists the setups used in the following simulations.

5.2.2 Experiment 1: Fully coordinated model

In this experiment, the fully coordinated model adopts the divide-and-conquer algorithm with three mobile sensors. That is dividing the AOI into three equally subareas and assigning each mobile sensors sweeping its subarea in parallel way. In order to avoid scanning gap arising from inaccuracy path tracking, the scanning path width does not make full use of detection sensor range $4m(2 \times 2m)$, setting as 3m instead. A snapshot of the sweep scenario is shown in Fig.5.2

AOI	Rectangle, $35m \times 30m$
Mobile sensor	KheperaIII, No. of sensors $= 3$, line speed $v = 0.5m/s$, Detection sensor range $r = 2m$, Neighbourhood radius $Nr = 4m$.
Targets	1000 static targets, uniformly distributed
Executions	5 runs of 1000 seconds each

Table 5.1: Simulation environment setups

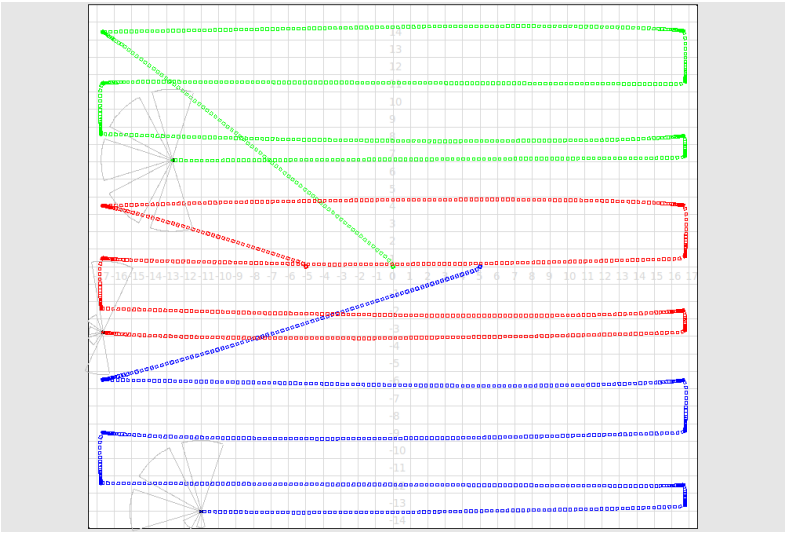

Figure 5.2: Fully coordinated model-sweeping of three mobile sensors (each sensor is represented by one color).

5.2.3 Experiment 2: Fully random model

As described in chapter 3, each mobile sensor moves to a direction that is random even distributed among $[0, 2\pi)$ by adopting fully random model. Thus, there is no task planning and assignment and only obstacles avoidance ability is needed for each sensor. Fig.5.3 shows simulation scenario of three mobile sensors exploring the AOI with fully random model.

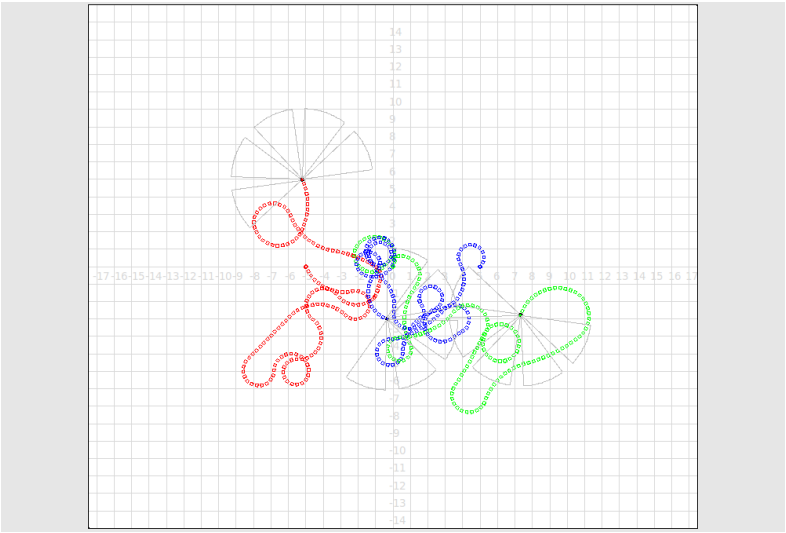

Figure 5.3: Fully random model.

Inherently, the fully random model has great environment adaptability because of the feature of no predefined path planning and obstacles avoidance. The Fig.5.4 demonstrates a scenario of mobile sensors covering the obstacle-filled AOI without any extra efforts.

5.2.4 Experiment 3: Anti-flocking model

As proposed in chapter 3, anti-flocking model works basing on neighbourhood awareness and coordinating sensors' cooperation with self-organizing rules that scatter around the AOI and explore unvisited area. Fig.5.5 demonstrates a scenario of anti-flocking model. Fig.5.6 also shows the scenario by adding obstacles.

5.2.5 Comparative Study

The following part compares area coverage and detection effectiveness of the above three models, with same setting of three mobile sensors.

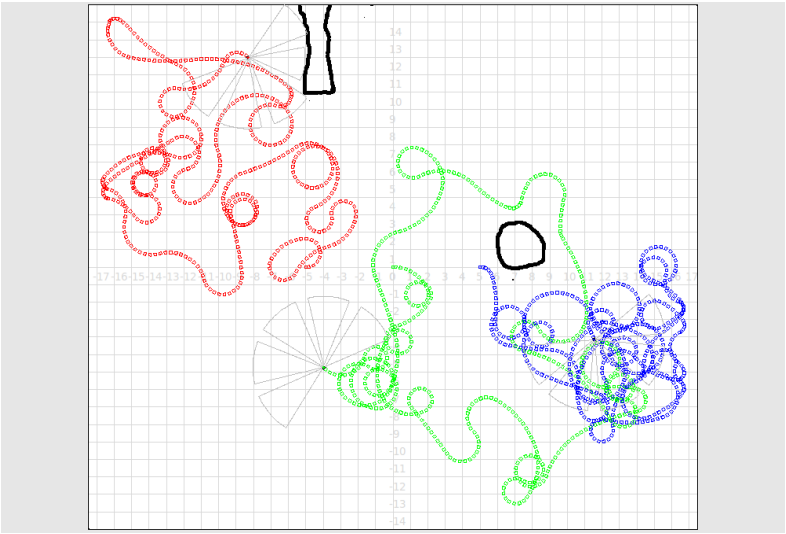

Figure 5.4: Fully random model in obstacles environment.

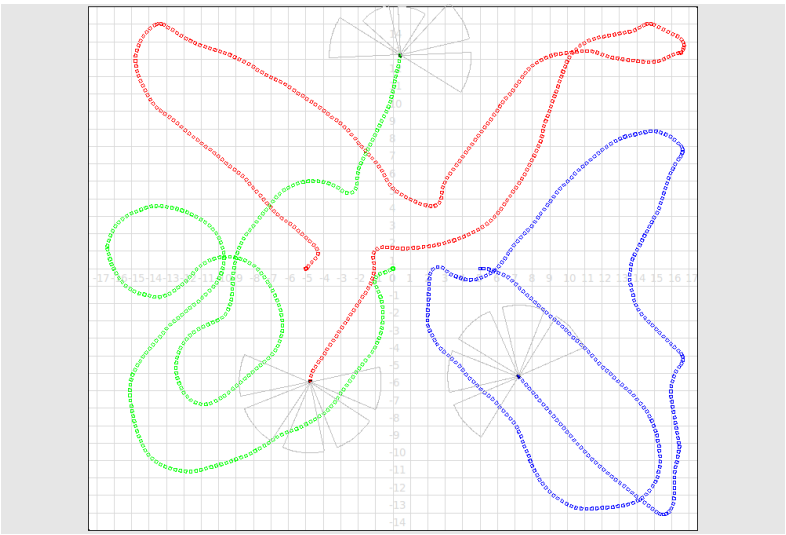

Figure 5.5: Anti-flocking model.

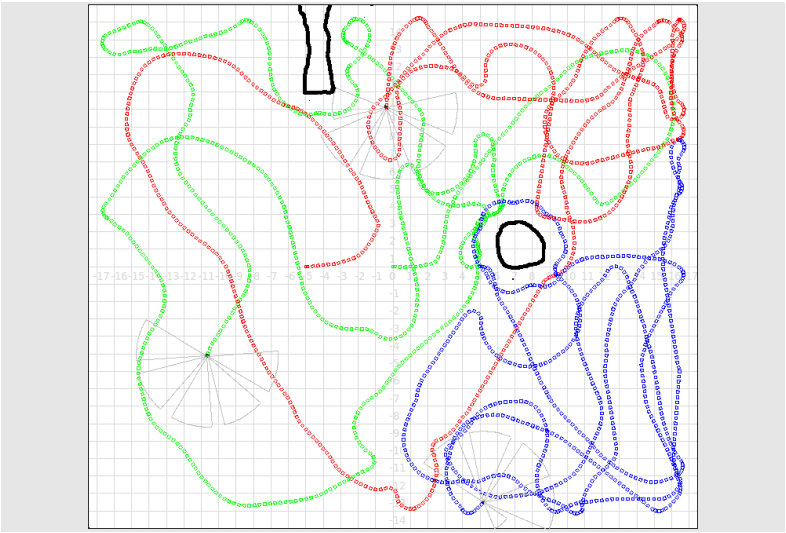

Figure 5.6: Anti-flocking model in obstacles environment.

Cumulative Area Coverage

Fig.5.7 shows the average of five runs of the cumulative area coverage results of three mobility models when three mobile sensors are used.

As shown in Fig.5.7, the cumulative area coverage increases as time going for all three models. However, the cumulative area coverage of random searching model increases at a very low rate, while the sweep searching model increases at almost a linear rate and reach to cover 100% of whole area of interest. The cumulative area coverage of anti-flocking model has the best covering in the beginning stage. Then, it turns flatter and follows a trend to cover all AOI.

Instantaneous Area Coverage

Table 5.2 shows the average instantaneous area coverage when three mobile sensors are used with different mobility models.

Instantaneous Area coverage is an important indication for applications that need monitoring interested area simultaneously. Comparing the instantaneous area coverage of these three mobility model, the anti-flocking makes best use of all mobile sensors' coverage because its de-centering rule leads to avoid coverage overlapping among the sensors. The random searching model's instantaneous area coverage is the second best because sensor nodes have less chance of coverage overlapping if fully randomly model applying in a large area with sparse sensor nodes, which is the case of this experiment. The observation that

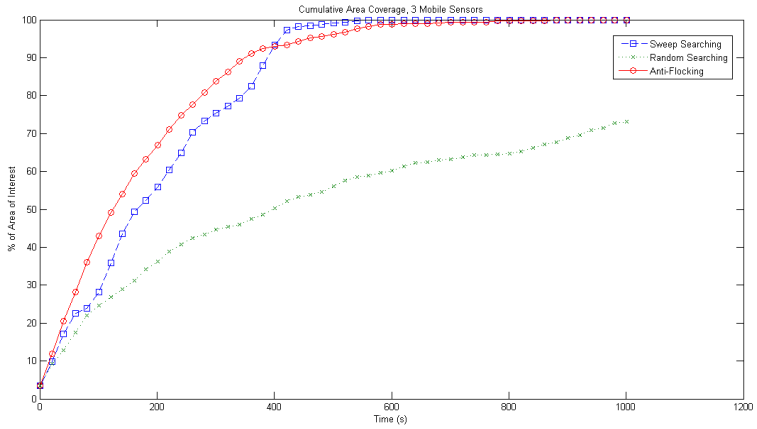

Figure 5.7: Cumulative area coverage results of three mobility models with 3 mobile sensors, average of 5 runs.

Mobility Model	Average Instantaneous Area Coverage
Fully Random Model	3.4383%
Fully Coordinated Model	3.3017%
Anti-Flocking Model	**3.4924%**

Table 5.2: Comparison of Instantaneous Area Coverage of 3 mobile sensors that applying three mobility models.

fully coordinated model has poor performance in the instantaneous area coverage results from path planning focusing on covering all corner area with sacrificing sensing range usage.

Events Detection

Table 5.3 summarizes the results of the events detection when three mobile sensors are used with different mobility models and 1000 injected events.

Mobility Model	Detection Rate	Ave. Detect Time (s)	Max Detect Time (s)
Fully Random	73.2%	303.91	N.A.
Fully Coordinated	**100%**	191.21	**551**
Anti-Flocking	99.9%	**163.54**	N.A.

Table 5.3: Comparison of Events Detection Efficiency of 3 mobile sensors that applying three mobility models.

As shown in Table 5.3, in term of detection rate the fully coordinated sweeping searching delivers 100%, while anti-flocking model is 99.9%, i.e. just 1 event undetected among totally 1000 events. In the contrast, the random searching model has the least detection rate, 73.2%.

In term of average detection time, the anti-flocking model has the shortest average detection time. The detection time of sweep searching has a close performance as anti-flocking. The average detection time of random searching model takes 1.86 times of that of the anti-flocking model.

Among the three models, only sweep searching model has a guaranteed maximum detection time. Both random searching and anti-flocking missed some events with given time out, 1000 seconds in this experiment setup.

These detection results reflect that the sweep searching model gain good performance and guaranteed quality of service (the worst case is under control) because of its carefully task planning and assignment. On the other hand, random searching is the worst model in all measures of detection because of its repetitive area searching and the lack of cooperation between the mobile sensors. The anti-flocking model gets the shortest average detection time, and much closed detection rate as that of sweeping model. The self-organization rules of anti-flocking model allow exploration of unvisited area and cooperation without explicit communication (stigmergic cooperation) among the mobile sensors by scattering them in the area of interest.

5.3 Experiments with real platform

To further explore the discussed algorithms' performance and features, several experiments using a real platform were carried out.

In the implementation, three KheperaIII mobile robots with 5 sonar sensors and 11 infrared sensors are used. Player server is running on the mobile robots platform. More details on the KheperaIII and its software environment and agent architecture are given in the appendix B.

5.3.1 Experiment 4: Multi-room platform and point scanning

A platform that mimics Waterloo International Airport was built in size of $144cm \times 123cm$, as shown in Fig.5.8. Fig. 5.9 is the rooms' layout of the airport.

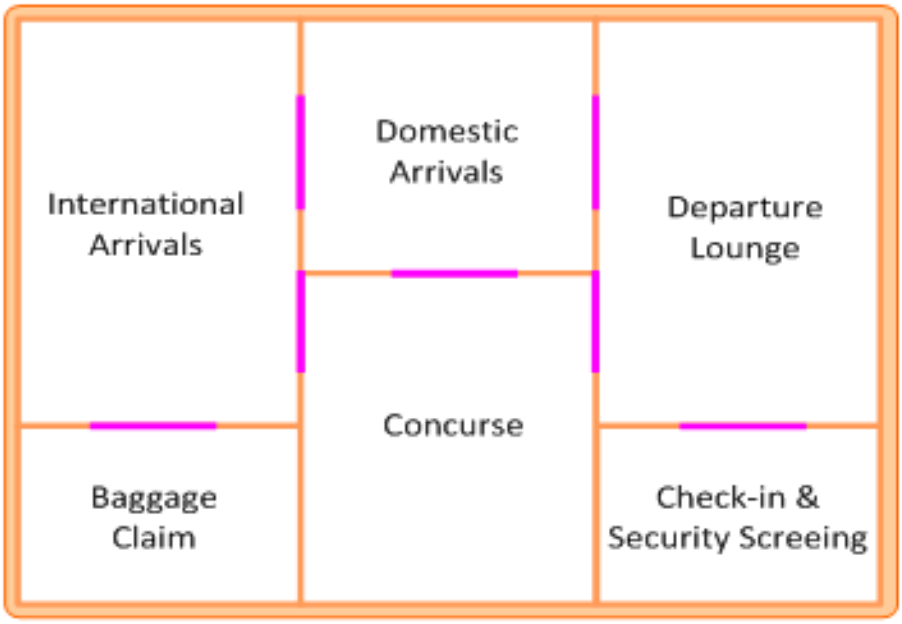

Figure 5.8: A platform mimics Waterloo International Airport.

In order to examine the multi-agent implementation system, one mobile robot demonstrates a process of scanning pre-defined key points in each room of the airport model, referred the on-line released video in [44]. Fig.5.10 is a snapshot of the points scanning in the airport model.

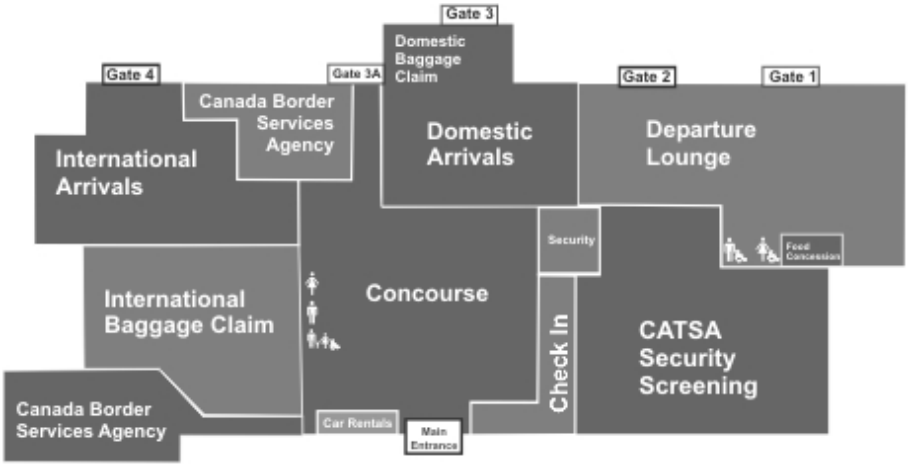

Figure 5.9: Layout of Waterloo International Airport (from the website of Waterloo Airport [43]).

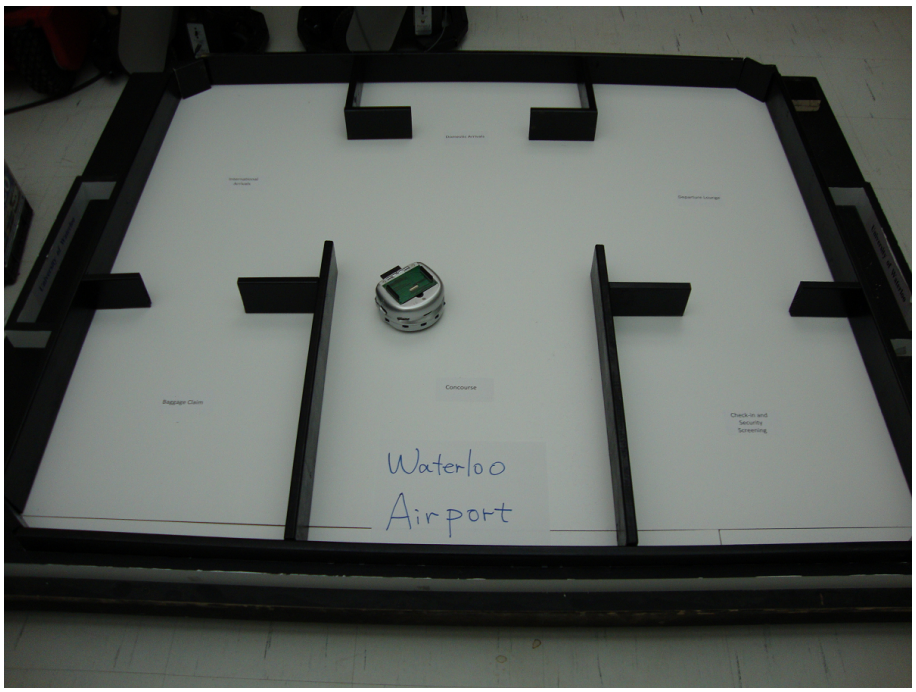

Figure 5.10: A snapshot of points scanning in the airport model.

5.3.2 Experiment 5: Executions of real platform on one room model

In order to have enough space to demonstrate algorithms working with three KheperaIII, one single room model based on the $144cm \times 123cm$ platform was setup as experimental area.

5.3.3 Parameters setup of real mobile sensors

Because relatively small space ($144cm \times 123cm$), the parameters are scaling down too as shown in Table 5.4.

AOI	Rectangle, $144cm \times 123cm$
Mobile sensor	KheperaIII, No. of sensors = 3, line speed $v = 0.1m/s$, Detection sensor range $r = 10cm$, Neighbourhood radius $Nr = 30cm$.
Targets	Simulated 100 static targets, uniformly distribution in AOI
Executions	500 seconds

Table 5.4: Parameters setting with real mobile sensors.

5.3.4 Results of executions in real platform

The collected videos demonstrated scenarios of the three mobility models, namely, fully coordinated model, fully random model, and anti-flocking model are released online at [44]. Fig.5.11 is a snapshot of the experiment.

Fig.5.12 illustrates cumulative area coverage of three mobility models.

Table 5.5 is the result of events detection.

The results of experiments on real platform reflect the same trend as the simulations. That is fully coordinated model achieved best performance in both area coverage and detection effectiveness. The fully random model performs the lowest in all metrics. The anti-flocking model achieved improved performance than fully random strategy.

Another observation is that all the recorded results perform lower than the simulations. The inaccuracy of platform measurement and space and time limitations are the main reasons resulting in the situations.

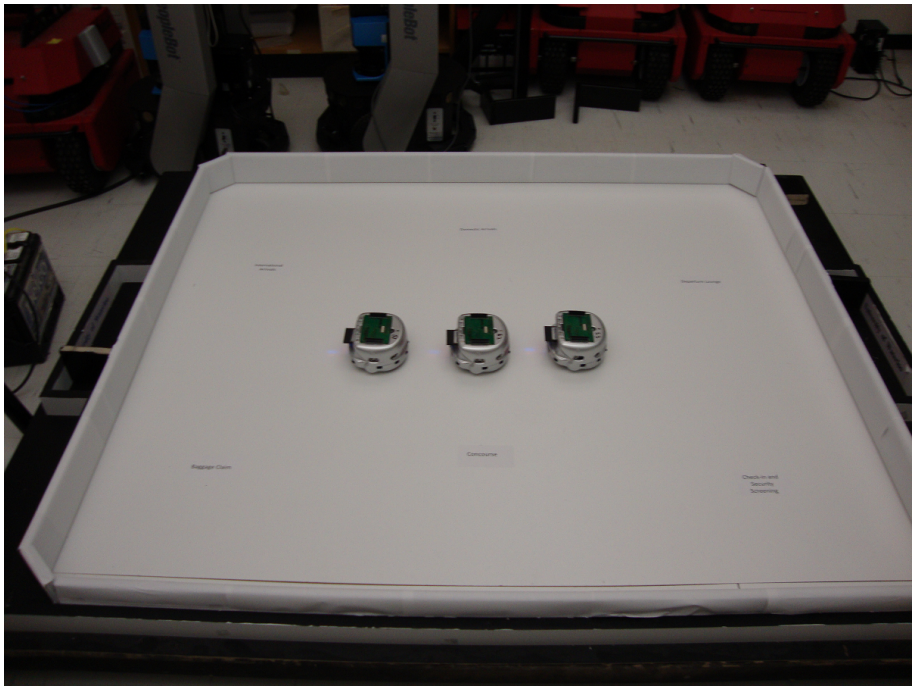

Figure 5.11: A snapshot of the implementation with real platform.

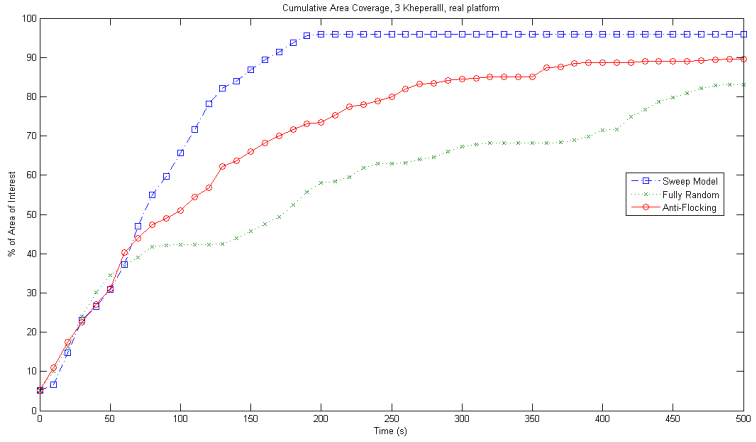

Figure 5.12: Cumulative area coverage of the three mobility models implemented with real platform.

Mobility Model	Detection Rate	Ave. Detect Time (s)	Max Detect Time (s)
Fully Random	83%	154.93	N.A.
Fully Coordinated	**96%**	**76.62**	N.A.
Anti-Flocking	90%	109.16	N.A.

Table 5.5: Comparison of Events Detection Efficiency of the three mobility models implemented with real platform.

5.4 Discussions

From the results of both simulation and real platform, fully random mobility, fully coordinated mobility and emergent mobility models are examined as guidance strategies for a set of mobile sensors searching for randomly distributed targets in mobile surveillance systems. It has been found that the area coverage and the detection effectiveness of these mobile sensors vary with the mobility model. Fully coordinated mobility model is the most efficient way in terms of area coverage and detection effectiveness. This superiority comes from a carefully task organization and assignment. However, this model has the following drawbacks:

- Non-Scalable: It is not suitable for large scale system with huge amount of mobile agents, because it depends on heavily communications between agents ($O(n^2)$), where n is the number of sensor nodes;
- Non-adaptable with dynamic environments: Searching task assignments need a predefined environment map, which is not available or dynamically changing in some applications;
- Non-robustness: System will need re-configuration when nodes failing or new nodes adding occur.

The fully random model is clearly not an efficient choice because of repetitive searching of each sensor and among sensors. However, the following advantages make it standout in the discussed specific circumstance:

- It is an easy and simple algorithm to implement. This feature results in applications to deploy "swarm robots", a large number of low cost mobile sensors;
- System level functions and performance requirements can be achieved by highly nodes redundancy. Thus, it is robust to nodes failure;

- Inherently, the model adapts to dynamic environment because it works without needing prior map and path planning;
- Agile targets cannot make use of its observations or knowledge to predict sensors' path and thereby evade detection because mobile sensors under random model are unpredictable.

The proposed anti-flocking model solves these problems of the fully coordinated model, whilst gaining great efficiency improvement comparing with the fully random model. The main advantages of this model can be summarizes as follows:

- Scalable: Even system units are in huge number, the anti-flocking algorithm works on broadcasting and neighborhood communication, that keeps communication and computation load within $(O(d^2))$, where d is the number of neighbors;
- Adaptive with dynamic environments: anti-flocking is a kind of self-organization rule-based group cooperation, which can adapt itself in dynamic environments. It works even in circumstance of no predefined environment map, which is usually not available or dynamically changing in some surveillance scenarios, such as battle fields;
- Robustness: System works seamlessly when part of nodes fail, although it may be degraded in performance. Also, system can autonomously re-organize itself when adding new nodes, which leads to improve system performance.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

This research work has revealed the literature on surveillance operations and the state-of-the-art of distributed mobile surveillance systems that encompassed mobile sensors.

While mobile surveillance systems bring extra performance and functionality benefits to surveillance operations, the problem of coordination of multiple mobile sensors gives us a new challenge. This thesis focused on the problem of how to optimally organize multiple mobile sensors to achieve certain system level goals.

A multi-agent platform to organize mobile sensor nodes, control nodes and actor nodes was implemented. It demonstrated great flexibility and was favourable for its distributed, autonomous and cooperative problem-solving characters.

Three categories of mobility control models, i.e. fully coordinated model, fully random model, and emergent control model, were studied. The relationship between mobile sensor's mobility models, area coverage and detection efficiency were analyzed and examined through extensive experiments of both simulations and real platform. The advantages and constraints of different algorithms were well studied.

A self-organizing algorithm named anti-flocking which mimics solitary animal's social behaviour was first proposed. Experimental results demonstrate its attractive performance improvement while providing desirable features such as scalability, robustness and adaptivity.

A new version of Java client interface library (to be released as free GNU software) was developed. It bridges present version of mobile robot control software package (Player/Stage), and will benefit both academic and industrial world.

6.2 Publications resulting from this work

Two peer reviewed conference papers were published under the processing of this research as listed below. And, we are planning to prepare a journal paper to summarize the work of this thesis.

- Yun-Qian Miao, Alaa Khamis, Mohamed Kamel, "Coordinated Motion Control of Mobile Sensors in Surveillance Systems," Special Session on Sensing and Perception, International Conference on Signals, Circuits and Systems (SCS'09), Djerba, Tunisia, 2009.
- Yun-Qian Miao, Alaa Khamis, Mohamed Kamel, "Applying Anti-Flocking Model in Mobile Surveillance Systems," Special Session on Distributed Surveillance Systems, International Conference on Autonomous and Intelligent Systems (AIS 2010), June 21-23, 2010, Povoa de Varzim, Portugal.

6.3 Future directions

As a future work, we will consider examining the scalability of the three mobility models by increasing the number of mobile sensors in the system and analyzing its performance and limitation.

Also, we will expand our work by extending the validation scenarios by considering the factor of moving targets. The mobility model for targets such as random motion, line motion, circle motion, etc. and its effect on the studied measurement of effectiveness in the mobile surveillance system is worthy of exploiting deeper.

Another direction is using more realistic detection models, such as vision system with PTZ camera, temperature sensor, etc. and studying more sophisticated data fusion algorithm that can combine data from different modalities in order to improve the accuracy of the estimate.

Appendix A

Implementation Software Environment

This appendix briefly introduces the software packages which are used in our implementation system and describes the installation procedures of them. In the section concerning a software package of JavaClient, version compatible problem is discussed and new developed version JavaClient3 is described.

A.1 Architecture of Software Environment

In the implementation system, JADE (Java Agent Development Framework) [16] is used as agent management platform. The mobile sensor's driver and simulation is constructed based on Player/Stage/Gazebo [60]. The Java agent communicates with Player/stage/Gazebo through a JavaClient [10] library.

Fig.A.1 illustrated the system architecture and software packages that were used.

A.2 Player/Stage/Gazebo as a Robot Driver and Simulation Environment

- **Player:** it is a hardware abstraction layer that is implemented in software and lies between the robot's hardware and software. So, basically Player is a set of interfaces that gives an access to the robots hardware. It gives us the ability to send, receive commands and information from robots' sensors and actuators.

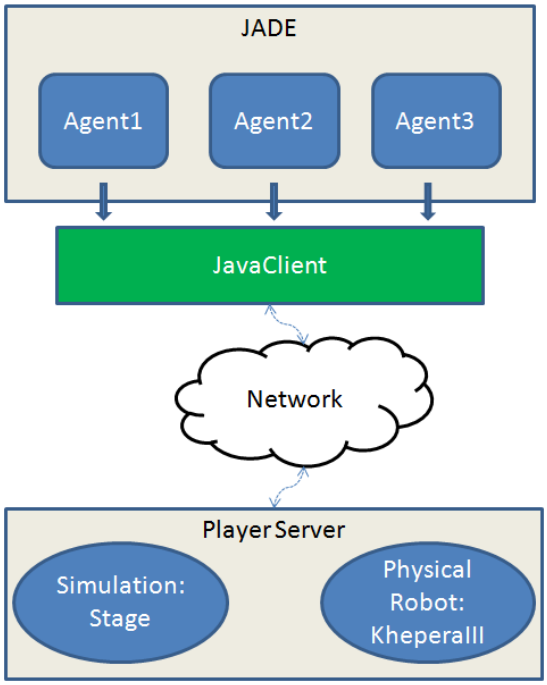

Figure A.1: Implementation Software Environment.

- **Stage:** it is a 2-D robot simulator (Fig.A.2) that simulates robots and its sensors and actuators like: laser, sonar devices and grippers in a 2-D bitmapped environment. Because Stage was originally developed to support the researches in the multi-robot systems, it has the ability to simulate a very large number of robots. So it is called a low quality high quantity simulator.

- **Gazebo:** is a multi-robot simulator for outdoor environments. Like Stage, it is capable of simulating a population of robots, sensors and objects, but does so in a three-dimensional world (Fig.A.3). It generates both realistic sensor feedback and physically plausible interactions between objects (it includes an accurate simulation of rigid-body physics).

Both Stage and Gazebo offer plugins to the Player, so, they can provide Player with virtual robots that we can write client control programs for them and connect it to the Player server. So, when those programs are used together they give us the ability to create and control simulated robots and test them against simulated environments and actions. By this, we will make sure of the system's correctness before installing it on real robots; also, the P/S/G will be useful in testing multi-robot cooperative behaviors that require multiple robots to run. So it will help us to avoid the failures that might happen and cause the robots to be damaged. And what is very nice in client programs written to control

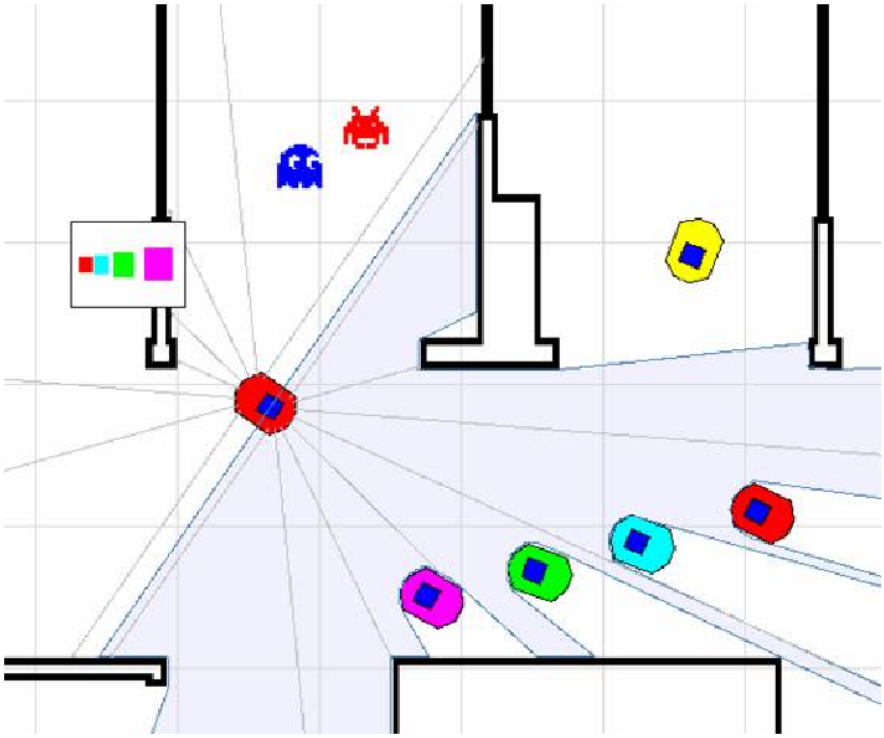

Figure A.2: Stage

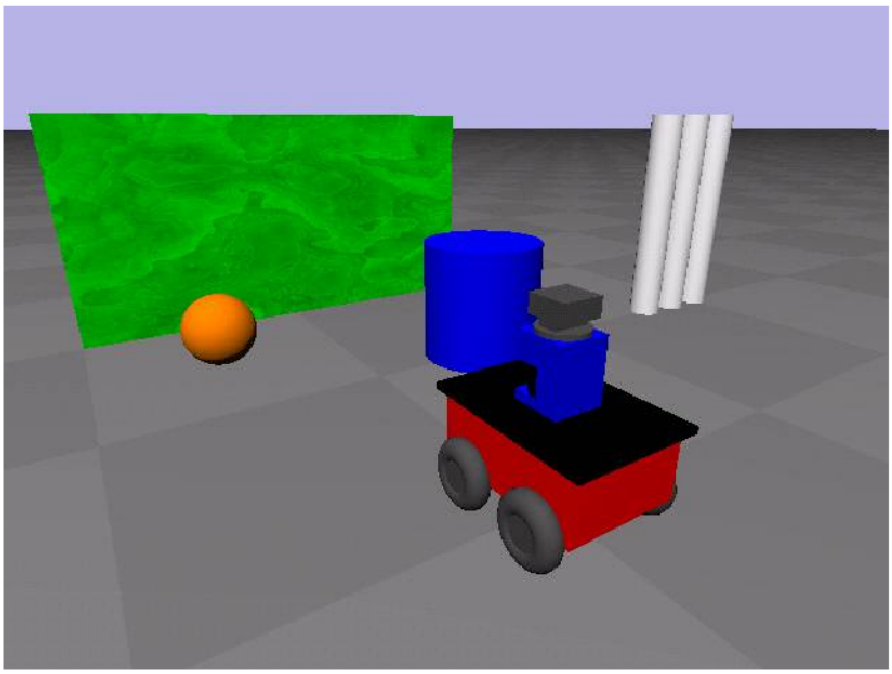

Figure A.3: Gazebo

simulated robots in P/S/G, that, we are able to transfer them to real robots with minor changes or even without any changes.

A.2.1 Installation of P/S/G

Because proposed system intended to use Player with Stage and Gazebo, we have to install the Player first and make sure that it is working before installing S/G. In this Appendix we will discuss the installation steps of each.

Player

1. Go to the SourceForge website and download Player source tarball version 2.1.2.
2. Uncompress the tarball. (The uncompressing command differs according to the distribution of Linux; uncompress it either from the command line or using the GUI).
3. Navigate directories using 'cd' command until get in the directory uncompressed the Player in.

4. To use the default configuration for the Player use: ./configure

5. Now, the compiling time; type: ./make

6. Finally, to start the installation process type: ./make install

During the installation process the system might ask for any additional packages and this is according to the distribution and installation of Linux. So, it has to be installed and re-run the Player installation from the beginning.
Now the Player is installed and ready to be used. It is installed in /usr/local by default. The pkg-config files must be added to system's path, so, you can compile programs that use player's libraries. This process should be done automatically, but, in some Linux distributions it does not. It can be added by type:

```
$ export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:$PKG_CONFIG_PATH
```

For more information regarding the Player installation, visit its webpage on SourceForge:
http://playerstage.sourceforge.net.

Stage

It is exactly similar as the Player:

1. Go to the SourceForge website and download Stage source tarball version 2.1.1.

2. Uncompress the tarball. (The uncompressing command differs according to the distribution of Linux; either do it from the command line or using the GUI).

3. Navigate to directories using 'cd' command until get in the directory you uncompressed the Stage in.

4. To use the default configuration for the Stage use: ./configure

5. Now, the compiling time; type: ./make

6. Finally, to start the installation process type: ./make install

Also, during the installation process the system might ask for any additional packages and this is according to distribution and installation of Linux. So, it has to be installed and re-run the installation from the beginning.
Now the Stage is installed and ready to be used. It is installed in /usr/local by default.

In the new versions of Stage (v.2.x.x), the Stage is no longer stand alone program. So, when using it with the Player, it provides a plug-in that adds the simulated robots to the Player. Hence, to run it by typing:

```
$ player <the configuration file name ".cfg">.
```

Gazebo

The installtion of Gazebo is a little bit tricky process because of the additional 3rd party software it needs. So, first, download and install the following software:

1. SWIG (Simplified Wrapper and Interface Generator): download it from its official website http://www.swig.org/.
2. wxPython (Python bindings for wxWidgets): its official website http://www.wxpython.org/.
3. ODE (Open Dynamic Engine): Library for simulating rigid body dynamics: http://www.ode.org/.

After that, proceed with Gazebo installation normally as follows:

1. Go to the SourceForge website and download Gazebo source tarball version 0.6.0.
2. Uncompress the tarball.
3. Navigate to directories using 'cd' command until get in the directory uncompressed the Gazebo in.
4. To use the default configuration for the Gazebo use: ./configure
5. Now, the compiling time; type: ./make
6. Finally, to start the installation process type: ./make install

Now start using Gazebo in two different modes:
either running the server only with out GUI:

```
$ gazebo <the world file name ".world">.
```

Or by having it running in the graphical mode:

```
$ wxgazebo <the world file name ".world">.
```

A.3 JADE as an Agent Development Platform

JADE (Fig.A.4) is a middleware that helps in developing multi-agent systems. It is consisted of three main parts:

- **Run Time Environment:** This is where the agents are created and it must be active on a given host. Each instance of it is represented with a container (because it might contain many agents). There has to be a main container initiated with the platform and all other possible non-main containers have to register to it.
- **A Library:** This includes the set of classes that is used to develop a multiagent system.
- **Graphical tools:** It helps managing the agents through a suit of graphical user interface tools. Operations like: create agents, kill, sniff, and others are done with a simple mouse click.

JADE was used as an agent development framework for this project because it facilitates the agent creation and managing process. And since JADE is FIPA compliant, it implements the FIPA standards and specifications. So, once the JADE is fired, the AMS (Agent Management System) and DF (Directory Facilitator) agents are created and start their jobs. The AMS will make sure that each agent will be created in its platform will be given a unique AID (Agent ID). Also, the AMS represents the high authority agent in the platform, in other words, it can create and kill other agents. The DF agent, however, acts as a yellow pages service for the agents; that is, agents can register their services with, so, other agents can inquiry them. Also, JADE provides the necessary infrastructure in order for the communications between agents take place. This communications are represented as sending and receiving ACL (Agent Communication Language) messages between agents. In addition, creating and initiating behaviors are done simply using JADE.

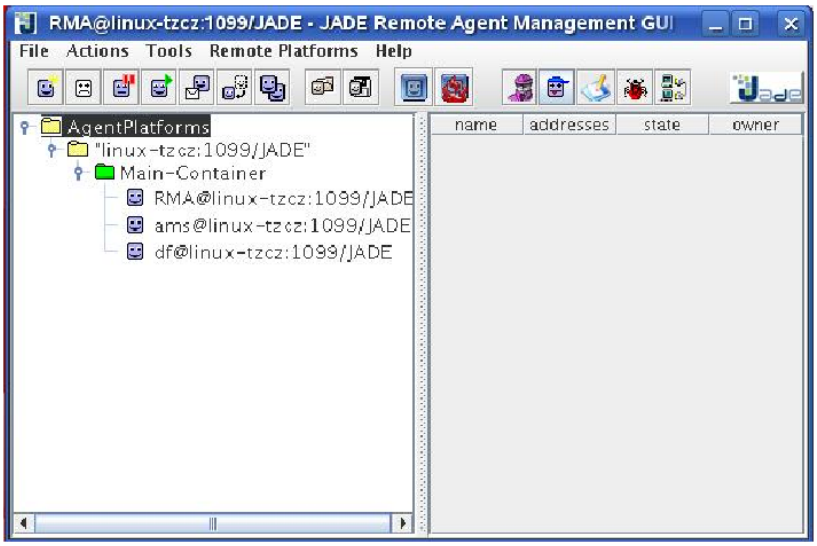

Figure A.4: JADE Platform

Other features in JADE include graphical tools that might be used for debugging purposes like:

- The dummy agent that is used to send messages to agents as in Fig.A.5.
- The sniffer agent that allows to inspect the communications between the agents as in Fig.A.6. It shows all the communications that happens between those five agents and the contents of any message can be viewed by clicking on it.

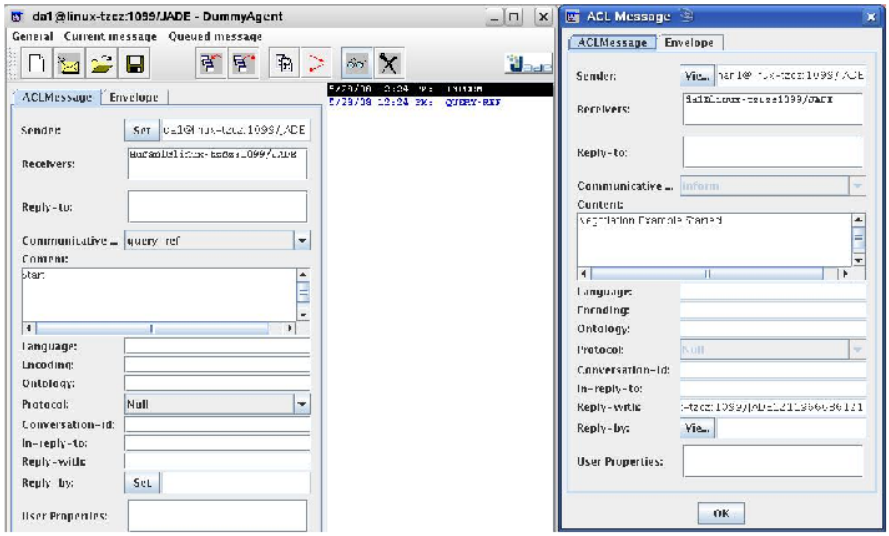

Figure A.5: The Dummy Agent

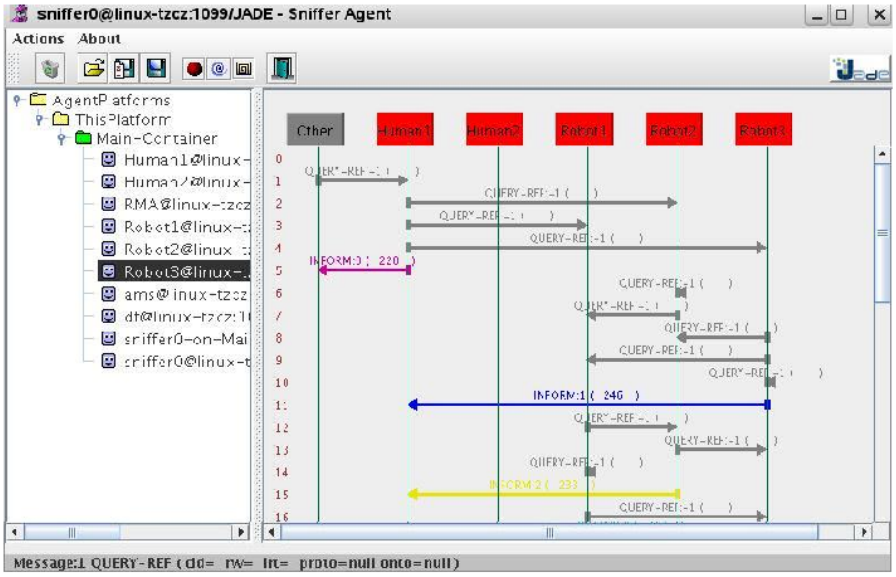

Figure A.6: The Sniffer Agent

A.3.1 Installation of JADE

JADE could be downloaded from its official website: http://jade.tilab.com/. It does not need any installation, only extracting the downloaded package. After that, it need to set the proper environment path.

```
export CLASSPATH=$CLASSPATH:/<Navigate to the path you extracted the JADE
in>/jade/lib/jade.jar:<JADE Path>/jade/lib/http.jar: <JADE Path>/jade/lib
/http.jar:<JADE Path>/jade/lib/iiop.jar: <JADE Path>/jade/lib /Base64.jar
```

Now in order to develop programs based on Agent-Paradigm, we have to have at least one java class that extends the Agent class and then type in the command line:

```
$ java jade.Boot <name of the agent>:<name of the class extending Agent>
```

A.4 JavaClient

JavaClient [10] library is a set of classes that implements all the player interfaces so it provides the capability of interacting with the player.

The latest version is JavaClient 2.0 from its website. It is compatible with jdk1.5. When trying to develop programs using player v.2.1.x, it does not work because it is compatible with Player v.2.0.x only.

A.4.1 A new bridge to Player v2.1.x: JavaClient3

The first version, named JavaClient, was designed to work with Player/Stage v1.6.x. The latest version, JavaClient2, was designed to work with Player/Stage v2.0.x. Even many research groups were looking for new version which can work with the latest version of Player/Stage, the JavaClient package was not updated since March 2006.

Because the real platform of robots, KheperaIII (explained in the next section), has the driver of Player v2.1.1 only, finding a solution to bridge our agent system to the real robots with Player v-2.1.1 is needed.

As a contribution, a new version named JavaClient3 package was developed and tested functionally through both simulation platform and real robots KheperaIII. The package is released at PAMI website [44].

Appendix B

Configuring the KheperaIII Robot

This appendix describes how to setup the real mobile robots, KheperaIII [**?**].

B.1 Overview of KheperaIII

The KheperaIII (Fig.B.1) is a miniature robot from K-Team Corporation, Switzerland. Features available on the platform can match the performances of much bigger robots, with an upgradeable embedded computing power using the KoreBot system, multiple sensor arrays for both long range and short range object detection, swappable battery pack system for optimal autonomy, and exceptional differential drive odometry.

Figure B.1: The KheperaIII Robot

The robot using the KoreBot, features a standard embedded Linux Operating System for quick and easy autonomous application development. It is easily interfaced with any

Personal Computer. Remote operation programs can be written with Matlab, LabView, or with any programming language supporting serial port communication.

The robot includes an array of 11 Infrared Sensors for obstacle detection as well as 5 Ultrasonic Sensors for long range object detection. It also proposes a front pair of ground Infrared Sensors for line following and table edge detection. The robot motor blocks use very high quality DC motors for efficiency and accuracy. The swappable battery pack system provides an unique solution for almost continuous experiments, as an empty battery can be replaced in a couple of seconds.

The robot is also able to host standard Compact Flash extension cards, supporting WiFi, Bluetooth, extra storage space, and many others.

B.2 Connect with KheperaIII

In order to communicate with KheperaIII, the connection can be established either by serial link or wireless. As an essential step, serial connection must be setup first because all other connections need have a prior connection to install the driver software.

B.2.1 Serial link by RS232

This configuration allows communicating between the KoreBotLE plugged on the robot and a host computer through a serial link. The host computer is linked to the interface module using a standard RS232 line. The adaptation RS232/TTL is made on the KoreBotLE. The following procedure describes how to use the serial communication mode:

1. Connect KoreConnect as Fig.B.2.
2. The charged Battery or a power supply is plugged, and the robot is turned ON.
3. The robot must be connected to a KoreConnect module using the serial cable.
4. The KoreConnect should be connected to the host computer using a RS232 non-crossed cable. In this mode, the cable has to be connected on the DB9 connector number 1 (see Fig.B.2).
5. Serial port configured as followed: 115200bps, 8 Data bits, 1 stop bit, no parity, no hardware control.
6. Host computer running a terminal software, such as Hyperterminal in Windows system.

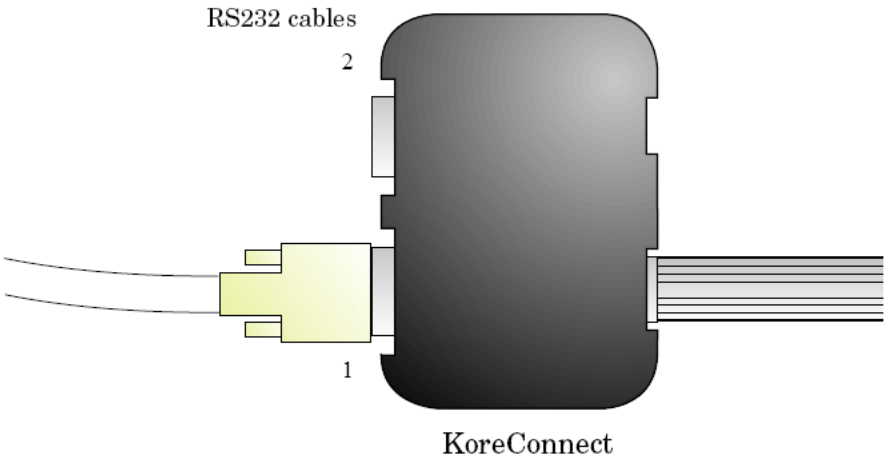

Figure B.2: Serial Connection with KheperaIII

B.2.2 Establish Wireless Connection

1. Insert a wireless compact flash card (Ambicom WL5400G-CF in our case) in the Korebot of KheperaIII.

2. Load the wifi module by typing:

   ```
   $ Modeprobe pxa2xx_cs>.
   ```

3. Configure the wireless network by modify the file /etc/network/interfaces with the designated wireless network settings:

   ```
   /***************** /etc/network/interfaces **********/
   # The loopback interface
   #
   auto lo
   iface lo inet loopback
   #
   # Wireless interfaces
   #
   auto eth0
   #iface eth0 inet dhcp
   iface eth0 inet static
   wireless_mode managed
   wireless_essid YOUR_SSID_OF_NETWORK
   ```

```
wireless key s:YOUR_PASSWORD_OF_NETWORK
address YOUR_IP_ADDRESS
netmask YOUR_NETMASK
gateway YOUR_GATEWAY_IP
/*******************************************/
```

4. Reboot the system or restart the network with the following command:

   `$ /etc/init.d/networking restart>.`

B.3 Installing and using Player on the KheperaIII

B.3.1 Required hardware and software

Required hardware:

- KheperaIII with KorebotLE
- A host computer and connection with KheperaIII(either serial or wireless connection)

Required software:

- linux operating system version 2.6.x on the host computer
- Player/Stage version 2.1 installed on the computer
- linux Kernel 2.6.x on the KorebotLE
- KheperaIII Player/Stage driver:
  - KheperaIII.cfg
  - KheperaIII.so
  - libltdl3_1.5.10-r3_armv5te.ipk
  - libstdc++6_4.1.2-r10_armv5te.ipk
  - player_2.1.1-r0_armv5te.ipk

B.3.2 Installation procedure

1. Power up the KheperaIII robot.
2. Establish a network connection with the KheperaIII either by wireless or the serial cable (see section 2 of this appendix).
3. Copy and install the following 3 packages on the robot:
   command for copying with ssh

   ```
   $ scp FILE root@KHEPERA_IP_ADDRESS:/home/root>.
   ```

   command for installation

   ```
   $ ipkg install PACKAGE_NAME>.
   ```

   packages files:

   - libstdc++6_4.1.2-r10_armv5te.ipk
   - libltdl3_1.5.10-r3_armv5te.ipk
   - player_2.1.1-r0_armv5te.ipk
4. Copy KheperaIII.so and KheperaIII.cfg on the korebot

B.3.3 Usage and Testing driving the robot

1. On the KoreBot, launch the server:

   ```
   $ player KheperaIII.cfg>.
   ```

2. On the host computer with Player 2.1.1 installed, export the library path and launch the viewer:

   ```
   $ export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH>.
   $ playerv -h IP_ADDRESS_OF_THE_ROBOT --ir:0 --position2d:0 --sonar:0 --power:0>.
   ```

3. Test driving the robot with the playerv interface (Fig.B.3):
   Go to "Devices/Position2d" and select "Command"
   ⇒ A red cross appears on the robot.
   Move this Red Cross to drive the robot.

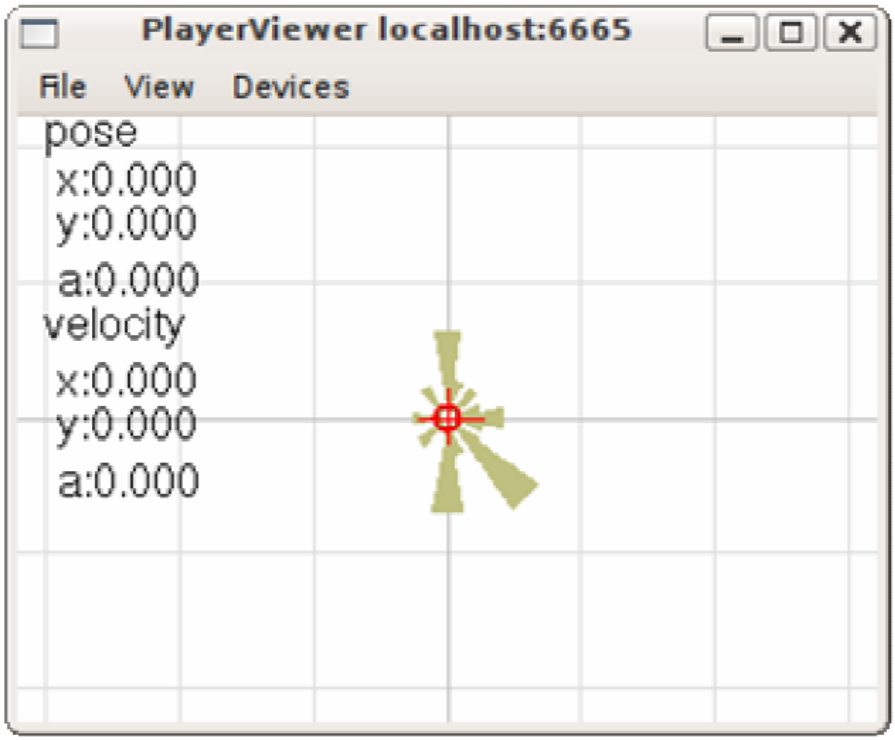

Figure B.3: PlayerViewer graphical interface

Appendix C

User Guide

C.1 Start the system

To use this framework, the first thing is to install Player/Stage/Gazebo programs and JADE as shown in Appendix A. Also, the needed JavaClient3 library can be downloaded from [44].

Second, KheperaIII robot should be configured as shown in Appendix B if system is going to run in real platform.

Finally, the system can be started by executing a batch file command. For example, the following batch file runs a system with JADE that includes three KheperaIII robots.

1. To start the player server by either simulation environment or real robot:

   ```
   $ player /<the path of .cfg file>/testbed.cfg
   ```

2. Execute the batch file command:

   ```
   $ ./TestMUSES_3Khepera
   ```

3. The batch file TestMUSES_3Khepera:

   ```
   $ java jade.Boot -gui &
   "R1:RobotAgent(IP1 PORT1 startX startY startYaw)" &
   "R2:RobotAgent(IP2 PORT2 startX startY startYaw)" &
   "R3:RobotAgent(IP3 PORT3 startX startY startYaw)" &
   "CTL:ControlAgent" &
   "Target:TargetModelAgent" &
   "LM:LogAgent"
   ```

C.2 Agents' acceptable commands

When the JADE platform starts with the robot agents, control agent, target modelling agent, and measurement log agent as mentioned above, the system works by sending ACL messages to the control agent, as shown in Fig.C.1. The acceptable commands of each agent are listed here:

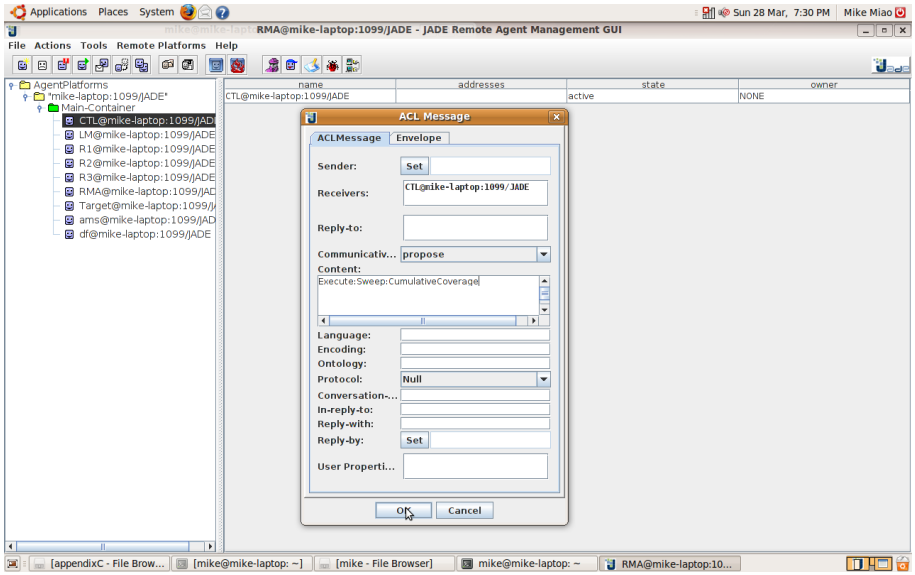

Figure C.1: System command through sending ACL message

C.2.1 Control Agent:

- *Initialize* : ⟨*parameter*1⟩
  - **Description:** System initialization by setting each mobile sensor to face a random selected orientation; and generating uniform distributed targets in the test scenario.
  - **parameter1:** the number of targets to be generated, refer the Target Modelling Agent.
  - **Example:** *Initialize* : 100
    All mobile sensors face a random direction, and generate 100 targets that uniform distributed in the scenario.
- *Execute* : ⟨*parameter*1⟩ : ⟨*parameter*2⟩

- **Description:** Organize mobile sensors following the mobility model given in parameter1; and log the measurement given in parameter2.
- **parameter1:** mobility model of mobile sensors, refer Mobility Control Agent.
- **parameter2:** measurement to be logged, refer Log Agent.
- **Example:** $Execute : Sweep : CumulativeCoverage$
  Mobile sensors follow "Sweep" mobility model, Log agent starts recording "CumulativeCoverage".

- *Stop*
  - **Description:** Ask mobile sensors to stop moving and Log agent stop recording.

C.2.2 Mobility Control Agent:

- $FullyRandom$
  - **Description:** ask mobile sensor to move using fully random model.
- $Sweep$
  - **Description:** ask mobile sensor to move using sweep model.
- $Antiflocking$
  - **Description:** ask mobile sensor to move using anti-flocking model.
- $Goto : \langle X \rangle : \langle Y \rangle : \langle Yaw \rangle$
  - **Description:** ask mobile sensor to move to the position given by parameter X, Y, and Yaw.
  - **parameter X:** destination position of coordinator x, in meter.
  - **parameter Y:** destination position of coordinator y, in meter.
  - **parameter Yaw:** destination facing orientation, in rad.
  - **Example:** $Goto : 0.5 : 1.2 : 0.8$
    Ask mobile sensor moving to the position $\langle 0.5m, 1.2m \rangle$ in 2D space, and the facing orientation is 0.8.
- $TeamFormation : \langle shape \rangle$
  - **Description:** ask mobile sensors to form a group, desired shape given by parameter *shape*.

- **parameter** *shape***:** the desired group shape, valid value can be: “Triangle”, “Line”.
- **Example:** $TeamFormation : Triangle$
  Ask mobile sensors forming a triangle shape.

- $Pointscan$
  - **Description:** ask mobile sensor to scan the predefined points of the test scenario.
- $StopMove$
  - **Description:** ask mobile sensor to stop moving.

C.2.3 Target Modelling Agent:

- $StartModelling : \langle parameter1 \rangle$
  - **Description:** Generate targets that uniform distributed in the test scenario.
  - **parameter1:** number of targets to be generated.
  - **Example:** $StartModelling : 100$
    Generate 100 targets that uniform distributed in the test scenario.
- $NoTarget$
  - **Description:** Not taking action of generating targets.

C.2.4 Log Agent:

- $StartRecord : \langle parameter1 \rangle : \langle parameter2 \rangle$
  - **Description:** Start recording measurement.
  - **parameter1:** type of measures to be recorded. Valid value: ”InstantCoverage”, ”CumulativeCoverage”, ”DetectionTarget”
  - **parameter2:** time length of measures recording, in second.
  - **Example:** $StartRecord : CumulativeCoverage : 1000$
    Start recording cumulative coverage for 1000 seconds.
- $StopRecord$
  - **Description:** stop the action of recording.

Bibliography

[1] B. D. O. Anderson, C. Yu, B. Fidan, and J. M. Hendrickx. Control and information architectures for formations. In *Proc 2006 IEEE Conference on Control Applications*, pages 1127–1138, 2006. 18, 22

[2] P. Avis. Surveillance and canadian maritime domestic security. *Canadian Military Journal*, 4:9–15, 2003. 8

[3] M. Batalin and G. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *6th International Conference on Distributed Autonomous Robotic Systems (DSRS02)*, 2002. 2, 15

[4] Abderrezak Benaskeur, Alaa Khamis, and Hengameh Irandoust. Cooperation in distributed surveillance. In *The International Conference on Autonomous and Intelligent Systems, June 21-23, 2010, Povoa de Varzim, Portugal.*, 2010. 31, 32

[5] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proc. 1997 Conf. on Computer Vision and Pattern Recognition, IEEE Computer Society*, page 495502, 1997. 8

[6] E. Bosse and J. Roy. The canada-netherlands collaboration on multisensor data fusion and other canada-nato msdf activities. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (ISCAS'98)*, 1998. 8

[7] G. Brassard and P. Bratley. *Fundamental of Algorithmics*. Prentice-Hall, 1996. 16, 30

[8] Mihaela Cardei and Ding-Zhu Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005. 34, 38

[9] Mihaela Cardei and Jie Wu. Coverage in wireless sensor networks. *Handbook of Sensor Networks*, CRC Press, 2004. 15, 31, 32, 33

[10] Java client for PLAYER website. http://java-player.sourceforge.net/, last accessed in Feb 2010. 43, 58, 67

[11] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja. Sensor deployment strategy for target detection. In *First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002. 1

[12] R. Cucchiara, C. Grana, A. Patri, G. Tardini, and R. Vezzani. Using computer vision techniques for dangerous situation detection in domotic applications. In *Proc. IEE Workshop on Intelligent Distributed Surveillance Systems*, pages 1–5, 2004. 8

[13] F. Dressler. *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons, Ltd, 2007. 10, 23

[14] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, 2000. 28, 36, 37, 38

[15] G.L. Foresti and L. Snidaro. *Multisensor Surveillance Systems: The Fusion Perspective*, chapter A Distributed Sensor Network for Video Surveillance of Outdoors, pages 8–27. Kluwer Academic Publishers, 2003. 6

[16] Java Agent DEvelopment framework website. http://jade.tilab.com/, last accessed in Feb 2010. 43, 58

[17] D.W. Gage. Command control for many-robot systems. In *Proceedings of Nineteenth Annual AUVS Technical Symposium*, pages 22–24, 1992. 2, 13, 16, 22, 34

[18] D.W. Gage. Randomized search strategies with imperfect sensors. In *Proceedings of SPIE Mobile Robots VIII*, volume 2058, pages 270–279, 1993. 2, 12, 13, 16, 22, 25, 26, 30

[19] Luc-Alain Giraldeau. *Behavioural Ecology, Part Three: Exploiting the Environment*. Oxford University Press, 2008. 27

[20] J. Heikkila and O. Silven. A real-time system for monitoring of cyclists and pedestrians. In *2nd IEEE Int. Workshop on Visual Surveillance, Colorado*, pages 74–81, 1999.

[21] L. Hodge and M. Kamel. An agent-based approach to multisensor coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(4):648–661, 2003. 2, 15, 35, 38

[22] A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *DARS 02*, 2002. 2, 15

[23] K. Ullas Karanth and Melvin E. Sunquist. Behavioural correlates of predation by tiger (panthera tigris), leopard (panthera pardus) and dhole (cuon alpinus) in nagarahole, india. *Journal of Zoology*, 250:255–265, 2000. 27

[24] S.P. Lalley and H.E. Robbins. Uniformly ergodic search in a disk. *Lecture notes in pure and applied mathematics*, 112:131–151, 1989. 26

[25] S. Li, C. Xu, W. Pan, and Y. Pan. Sensor deployment optimization for detecting maneuvering targets. In *7th International Conference on Information Fusion (FUSION)*, pages 1629–1635, 2005. 2, 15

[26] X.-Y. Li, P.-J. Wan, and O. Frieder. Coverage in wireless ad-hoc sensor networks. *IEEE Transactions on Computers*, 52:753–763, 2002.

[27] B. Liu, P. Brass, and O. Dousse. Mobility improves coverage of sensor networks. In *MobiHoc05*, 2005. 15, 22, 30, 31, 32, 36, 38

[28] B. Liu and D. Towsley. On the coverage and detectability of large-scale wireless sensor networks. In *Proc. of the Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks Conference (WiOpt)*, 2003. 31, 36

[29] B. Liu and D. Towsley. A study on the coverage of large-scale sensor networks. In *The 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004. 1

[30] Lei Liu, Yongji Wang, Shuanghua Yang, Graham Watson, and Brian Ford. Experimental studies of multi-robot formation and transforming. In *Proceedings of the UKACC International Conference on Control 2008*, page www.control2008.org/papers/p146.pdf, 2008. xi, 18, 20, 22, 30

[31] Benny Ping Lai Lo, Jie Sun, and S.A. Velastin. Fusing visual and audio information in a distributed intelligent surveillance system for public transport systems. *Acta Automatica Sinica*, 29 (03):393–407, 2003. 8

[32] David Lyon. *Surveillance after September 11.* Polity Press, 2003. 1, 5

[33] Y. Mao and M. Wu. Coordinated sensor deployment for improving secure communications and sensing coverage. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 117–128, 2005. ISBN:1-59593-227-5. 2, 15

[34] M.J. McNish. Effects of uniform target density on random search. Master's thesis, Naval Postgraduate School, 1987. 25, 30

[35] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proc. IEEE Infocom*, volume 3, pages 1380–1387, 2001. 2, 31, 32, 35, 38

[36] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad hoc sensor networks. In *Proc. of 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 139–150, 2001. 12

[37] J. Meguro and J. Takiguchi. Development of an autonomous mobile surveillance system using a network-based rtk-gps. In *IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain.*, 2005. 2

[38] Y. Miao, A. Khamis, and M. Kamel. Coordinated motion control of mobile sensors in surveillance systems. In *Proceeding of 2009 International Conference on Signals, Circuits and Systems*, 2009. 1, 11, 16

[39] C. Micheloni, G.L. Foresti, and L. Snidaro. A co-operative multicamera system for video-surveillance of parking lots. In *Intelligent Distributed Surveillance Systems Symp. by the IEE, London*, pages 21–24, 2003. 8

[40] T. Newman and S. Hayman. *Policing, Surveillance, and Social Control.* Willan Publishing, 2001. 1

[41] A. T. Nghiem, F. Bremond, M. Thonnat, and V. Valentin. Etiseo, performance evaluation for video surveillance systems. In *AVSS '07: Proceedings of the 2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 476–481, Washington, DC, USA, 2007. IEEE Computer Society. 31, 36, 38

[42] Center of Excellence for Battlefield Sensor Fusion website. http://www.tnstate.edu/ce-bsf/, Last access in March 2010. 8

[43] Region of Waterloo International Airport website. http://www.waterlooairport.ca/en/, last accessed in March 2010. xii, 51

[44] PAMI Lab (University of Waterloo) website. http://pami.uwaterloo.ca/, last accessed in March 2010. 50, 52, 67, 74

[45] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. on Automatic Control*, 51(3):401–420, 2006. 24

[46] R. Olfati-Saber. Distributed tracking for mobile sensor networks with information-driven mobility. In *Proceedings of the 2007 American Control Conference*, 2007. 8, 24

[47] J. Pearce, P. Rybski, S. Stoeter, and N. Papanikolopoulous. Dispersion behaviors for a team of multiple miniature robots. In *IEEE International Conference on Robotics and Automation*, 2003. 15

[48] Sameera Poduri and Gaurav S. Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation*, pages 165–172, 2004. 36

[49] MUSES_SECRET project website. http://129.97.86.45:88/redmine/projects/show/muses, Last access in March 2010. 1, 9

[50] C. W. Reynolds. Flocks, herds, and schools: a distributed behavioural model. In *Computer Graphics (ACM SIGGRAPH 87 Conference Proceedings)*, volume 21(4), pages 25–34, 1987. 24, 30

[51] P.E. Rybski, S.A. Stoeter, M. Gini, D.F. Hougen, and N.P. Papanikolopoulos. Performance of a distributed robotic system using shared communications channels. *IEEE Trans. Robot. Autom.*, 18(5):713727, 2002.

[52] S. Sakane, T. Sato, H. Maruya, H. Okoshi, N. Nakamura, and M. Kakikura. A decentralized and cooperative sensing system for robot vision. *IEEE Trans. Robot. Automat.*, 11:86–104, 1996. 2

[53] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In *Proc. IEEE Infocom*, 2003. 1

[54] Y. Tseng, Y. Wang, K. Cheng, and Y. Hsieh. imouse: An integrated mobile surveillance and wireless sensor system. In *Proceedings of the 8th ACM international Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (Montreal, Quebec, Canada, October 10 - 13, 2005)*, pages 178–181, 2005. xi, 6, 9

[55] M. Valera and S.A. Velastin. Intelligent distributed surveillance systems: a review. In *IEEE Proc.-Vis. Image Signal Process.*, volume 152, pages 192–204, 2005. x, 1, 6, 7, 11

[56] T.J. Valone and L.A. Giraldeau. Patch estimation by group foragers: what information is used? *Animal Behaviour*, 45:721–728, 1993. 27

[57] M.E. Weber and M.L. Stone. Low altitude wind shear detection using airport surveillance radars. *IEEE Aerosp. Electron. Syst. Mag.*, 10(6):3–9, 1995. 8

[58] K-Team website. Kheperaiii player/stage software driver, http://ftp.kteam.com/kheperaiii/player_stage, last accessed in Nov 2009. 43

[59] Pervasa Inc website. http://www.pervasa.com, Last access in March 2010. 8

[60] Player/Stage website. http://playerstage.sourceforge.net/, last accessed in Feb 2010. 43, 58

[61] M. Xu, L. Lowey, , and J. Orwell. Architecture and algorithms for tracking football players with multiple cameras. In *Proc. IEE Workshop on Intelligent Distributed Surveillance Systems, London*, pages 51–56, 2004.

[62] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12:317–370, 2003. 41

[63] Zhi-Hong Zeng. Lane detection and car tracking on the highway. *Acta Automatica Sinica*, 29(3):450–456, 2003. 8

[64] Honghai Zhang and Jennifer C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. In *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks*, 2004. 36, 38

[65] Zhigang Zhu and Thomas Huang. *Multimodal Surveillance: an Introduction*, chapter 1, pages 1–10. Artech House Publisher, 2007. 5

[66] Y. Zou and K. Chakrabarty. Distributed mobility management for target tracking in mobile sensor networks. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 6(8):872–887, 2007. 2, 13, 15